REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 30-06-2006	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* Dec 2001 - Mar 2006

4. TITLE AND SUBTITLE
Performance Assessment Tools for Distance Learning and Simulation

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00014-02-1-0179

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Allen Munro and Quentin A. Pizzini
USC Center for Cognitive Technology
David G. Brill and Joanne K. Michiuye
UCLA/CRESST

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UCLA CSE/CRESST
300 Charles E. Young Dr. North
300 GSE&IS/Mailbox 951522
Los Angeles, CA 90095-1522

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report describes research conducted by the UCLA National Center for Research on Evaluation, Standards, and Student Testing (CRESST) and its subcontractor, the University of Southern California Behavioral Technology Laboratories (BTL), on performance assessment tools developed by (a) extending CRESST's knowledge mapping tool's authoring and scoring functionality and providing the capability to embed a knowledge mapping assessment in simulation-based training developed by BTL. Products are described in detail including the knowledge mapping tool with authoring and scoring systems, performance-based assessments, an instructional simulation providing interactive training, and the performance-based assessments embedded in the instructional simulation.

15. SUBJECT TERMS
performance assessment tools, knowledge mapping, simulation-based training

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)*
Unclassified	Unclassified	Unclassified			

Performance Assessment Tools for Distance Learning and Simulation

Deliverable – June 2006

Knowledge, Models and Tools to Improve
the Effectiveness of Naval Distance Learning

Eva L. Baker
CRESST/University of California, Los Angeles

Office of Naval Research
Award # N00014-02-1-0179

National Center for Research on Evaluation,
Standards, and Student Testing (CRESST)
Center for the Study of Evaluation (CSE)
Graduate School of Education & Information Studies
University of California, Los Angeles
301 GSE&IS, Box 951522
Los Angeles, CA 90095-1522
(310) 206-1532

The work reported herein was supported under Office of Naval Research Award Number #N00014-02-1-0179, as administered by the Office of Naval Research. The findings and opinions expressed in this report do not necessarily reflect the positions or policies of the Office of Naval Research.

Table of Contents

PERFORMANCE ASSESSMENT TOOLS FOR DISTANCE LEARNING AND SIMULATION

Allen Munro and Quentin A. Pizzini

USC Center for Cognitive Technology

David G. Brill and Joanne K. Michiuye

CRESST/University of California, Los Angeles

Abstract

Performance assessment tools for distance learning and simulation were developed by (a) extending the UCLA Center for Research on Evaluation, Standards, and Student Testing (UCLA/CRESST) knowledge mapping tool's authoring and scoring functionality and providing the capability to embed a knowledge mapping assessment in simulation-based training developed by the University of Southern California Center for Cognitive Technology (USC/CCT). Products are described in detail, including the knowledge mapping tool with authoring and scoring systems, performance-based assessments, an instructional simulation providing interactive training, and the performance-based assessments embedded in the instructional simulation.

Introduction

In prior work, the UCLA Center for Research on Evaluation, Standards, and Student Testing (CRESST) has completed development and testing of assessment tools for distance learning and simulation, including an online assessment tool for measuring a learner's deep understanding of a knowledge domain by creating a graphical model or map of the concepts and procedures in the knowledge domain and their relationships. This tool has been applied to online assessment in USMC rifle marksmanship training and USN Engineering Duty Officer training, and with USMC funding has been transitioned for use in the USMC Weapons Training Battalion Rifle Marksmanship Coaches Course. The tool includes automated scoring to enable accurate, fast, reliable, and cost-effective online use, and an authoring system providing multiple formats for flexibility and cost savings. The quality of measures has been tested in reliability and validity studies, and CRESST has completed a pilot study of the technical quality of online knowledge mapping for distance learning in a Joint Special Operations context.

The goal of this project was to develop, test, and evaluate additional performance assessment tools for distance learning and simulation by (a) extending the mapping tool's authoring and scoring functionality for application to performance-based assessment of deep understanding in a Navy-relevant domain, and (b) providing the capability to embed the assessment in simulation-based training. CRESST produced an assessment authoring system that can be used to generate performance assessments integrated with an instructional simulation provided by our subcontractor, the University of Southern California Center for Cognitive Technology (USC/CCT).[1]

Products include the knowledge mapping tool with an authoring system, performance-based assessments produced by the system, an instructional simulation providing interactive training, and the performance-based assessments embedded in the instructional simulation. We first describe the knowledge mapping tool, then the authoring system, the assessments, and the simulation with the embedded assessments.

[1] CCT was formerly called the Behavioral Technology Laboratory at the University of Southern California, USC/BTL.

The Standalone Knowledge Mapping Tool (SKMT)

CRESST followed the development of its server-based knowledge mapping tool (the Human Performance Knowledge Mapping Tool, or HPKMT, as reported in Chung et al., 2002) with a standalone version—the Standalone Knowledge Mapping Tool, SKMT—that did not require an Internet connection. In most respects, the SKMT functions the same way as the HPKMT. Differences are outlined below.

User Login Identification

Upon start-up, the SKMT asks the user to enter a login ID. In this standalone server-less environment, there is no centralized password authentication authority. Therefore it is incumbent upon instructors or task administrators to ensure users are entering login IDs correctly.

Saving Map Files

SKMT is launched from "runMapper.bat," executing locally on the PC without requiring any network server interactions. Knowledge maps are saved in the "StandAloneKMData" directory bundled with the SKMT. Knowledge map file names are encoded with the user's login ID, the ID number of the knowledge mapping task (assigned by the Task Authoring Tool), and the map name when saved by the user. Map names in the SKMT are limited to 20 alphanumeric characters.

Exporting and Importing Knowledge Maps

The export and import menu commands in the Advanced menu of the SKMT are used to explicitly copy maps to and from the SKMT to file system media (for exchanging maps between different machines), but these functions are not likely to be used frequently. The menu items are disabled if the current drawing area contains a modified map that has not been saved yet. This is done to avoid confusion and possible errors arising from the user attempting to export an unsaved map or to import a map with a name that collides with an unsaved map.

Standalone Knowledge Map Task Authoring Tool (SKMTAT)

The authoring tool CRESST developed to create tasks to be delivered by the SKMT was designed to perform only the most basic functions. The Standalone

Knowledge Map Task Authoring Tool (SKMTAT) is a Java-based application launched by running "runAuthor.bat" that generates knowledge map tasks and stores them in the ".mkm.taskdefs" file.

When there are no tasks in the ".mkm.taskdefs" file, the SKMTAT window appears as in Figure 1 below.

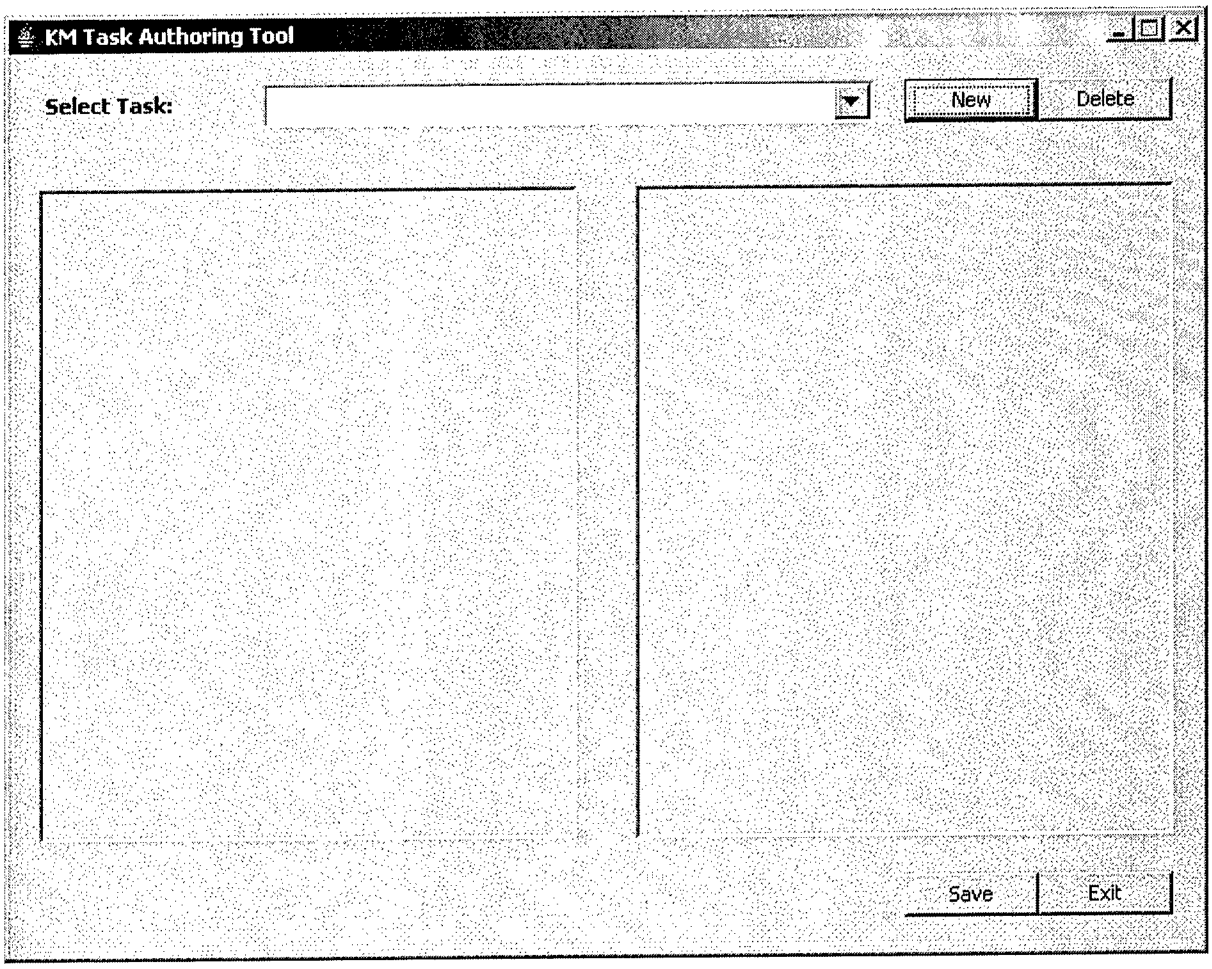


Figure 1. SKMTAT main window.

To enter a task, the user clicks the *New* button. The user is then prompted to enter a task name (see Figure 2).

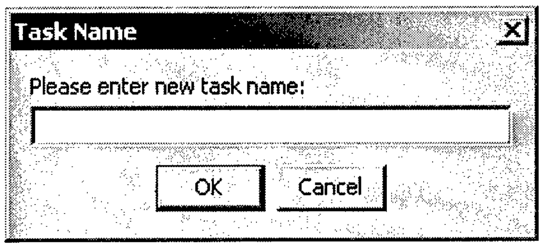
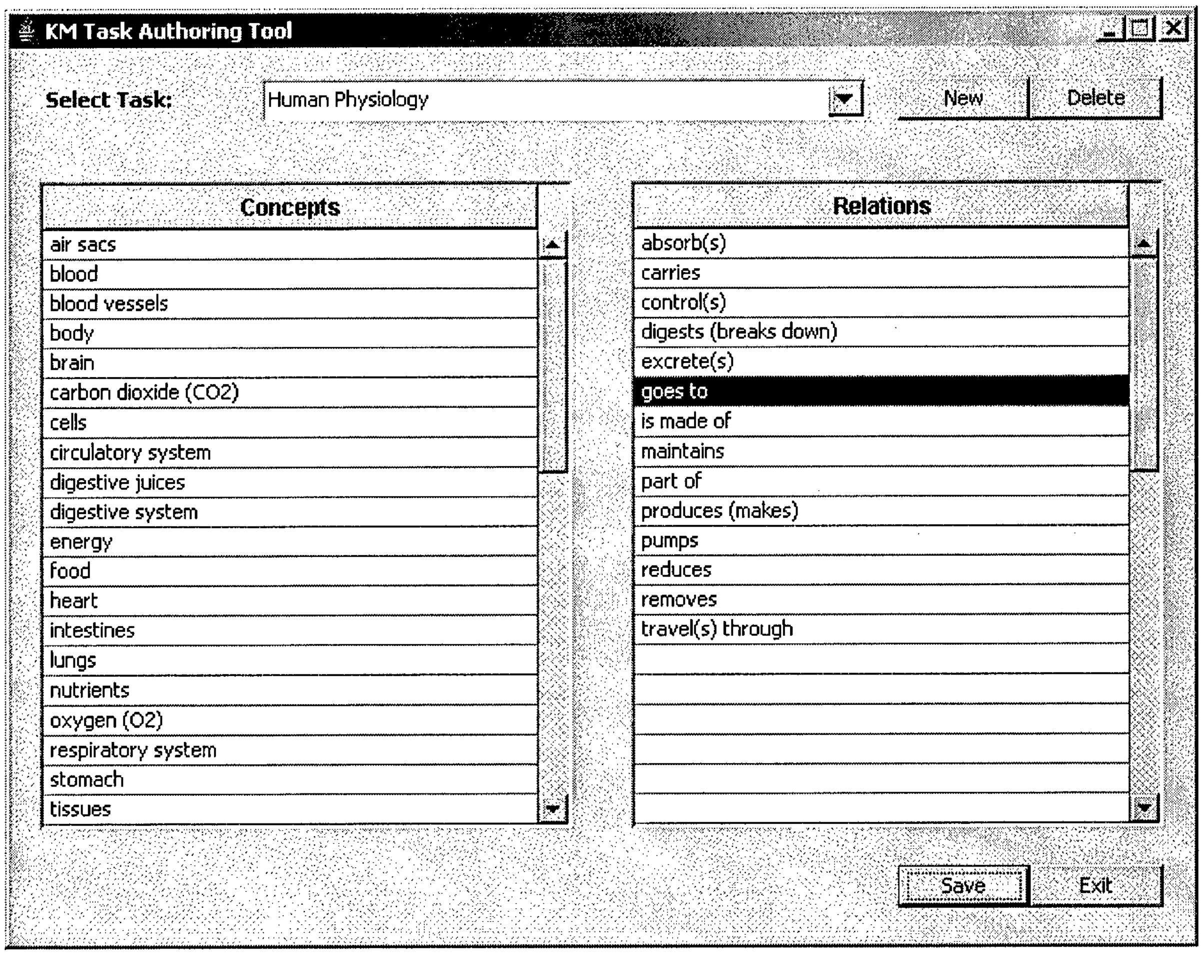


Figure 2. SKMTAT new task name dialog.

Concepts and relations for the new knowledge map task are entered in their respective columns. The lists are automatically alphabetized. To save a task, the user clicks the *Save* button at the bottom of the window. See Figure 3 for an example of an authored task.



Human Physiology

Concepts	Relations
air sacs	absorb(s)
blood	carries
blood vessels	control(s)
body	digests (breaks down)
brain	excrete(s)
carbon dioxide (CO2)	goes to
cells	is made of
circulatory system	maintains
digestive juices	part of
digestive system	produces (makes)
energy	pumps
food	reduces
heart	removes
intestines	travel(s) through
lungs	
nutrients	
oxygen (O2)	
respiratory system	
stomach	
tissues	

Figure 3. SKMTAT sample task.

When the task is saved, the SKMTAT assigns a task ID number to the task; this number is unique within the task definitions file, but not necessarily across different instances of the task definitions file if there are multiple authors authoring tasks, because this is a standalone, and not a server-based, system. If maintaining one set of knowledge mapping tasks at a site is of high importance, it is recommended that only one SKMTAT instance be in use, and updated task definitions files be disseminated to all SKMT applications on site.

Standalone Knowledge Map Scoring Tool (SKMST)

In CRESST's server-based authoring system for the HPKMT (described in Chung et al, 2004), automated scoring of knowledge maps was part of its functionality. In the standalone version of the authoring system, the scoring functionality is supplied by a separate application. The Standalone Knowledge Map Scoring Tool (SKMST) is a Java-based application that generates XML files for each student map scored.

Currently, expert maps are placed in a directory specifically for criterion maps. Accessible knowledge map tasks are listed in the SKMST. When a task is selected, applicable criterion maps are listed. The directory where student maps to be scored is designated in the SKMST, as well as the directory where score files should be stored.

The student score XML files have the student's login ID encoded in the file name to aid identification. The files can be imported into Microsoft Excel for data processing. Scores that are generated include the following:

- ***Exact*** scoring produces the number of propositions in the student map that exactly match propositions in that expert's map.
- ***Directionless*** scoring generates the number of propositions in the student map that match propositions in the expert map if you disregard the arrow direction but take into account the relationship label.
- ***Linkless with Direction*** scoring produces the number of propositions in the student map that match propositions in the expert map if you disregard the relationship label but take into account the arrow direction.
- ***Linkless*** scoring produces the number of propositions in the student map that match propositions in the expert map if you disregard both the arrow direction and the relationship label.

The Instructional Simulation

Goals and Objectives of the Research

To achieve the overall Capable Manpower FNC objectives, the objectives of the proposed USC/CCT research and development were integrated with the objectives that are the responsibility of the prime contractor, UCLA/CRESST. This supplement to the original KMT project supported the development of a simulation for use in learning Navy subject matter, new research on simulation authoring as an assessment methodology, and the integration of the CRESST Knowledge Mapper with iRides.

Technical Background

This section describes earlier work performed on the ONR Knowledge, Models and Tools to Improve the Effectiveness of Naval Distance Learning (KMT) project that provides a background for the proposed effort.

Learning Infrastructure. The University of Southern California's Center for Cognitive Technology (CCT) developed several simulation authoring technologies to support certain of the advanced distributed learning capabilities delivered in the KMT project. Some of these are outlined here.

- *Robust authoring and debugging of simulation behavior.* Simulation data editors for objects, attributes, and events were completed, and Copy and Paste capabilities were added to these editors. Methods for accessing behavior editors directly from the graphics were introduced. Simple graphics can be promoted to full behaving objects, and objects can be stripped of their rules and demoted to simple graphic status. A debugging interface was developed that supports stepping through the execution of a simulation. Invoked events can be stepped into or completed in one step, under user control. Simulation break points can be inserted. A set of features that support object cloning, including cloning under rule control, was implemented.

- *Enhanced Java 2D graphics.* Graphic objects can have transparency, and the degree of transparency can be controlled interactively by behavior rules. If an author assigns to the FillPattern attribute of a graphic object the file name of a jpg or gif image, that image will be used as a texture for the object. Authors were also given the facility to open modal and modeless

Java Swing dialogs under simulation rule control, making it possible to develop more professional-looking applications.

- *Optional SVG graphics.* The new Scalable Vector Graphics standard (SVG), which is receiving wide support in the application software industry, can now be used to develop certain types of iRides simulations. This makes it possible to use Corel Draw or Adobe Illustrator to develop simulation scenes. Behavior can then be added in the iRides Author application.

- *Improved data storage and retrieval.* A simple and open *Records Broker* interface was developed for iRides. In addition, methods for carrying out data storage and retrieval using the CRESST database system were implemented. This made it possible for a lesson to make use of CRESST assessment data to determine which instructional options to make use of during training.

- *Advances in iRides instructional capabilities.* The LML markup language and the tutorial controller that interprets it were significantly upgraded to support new capabilities, including making use of several simulations in a single lesson. Improved Question-and-Answer interfaces were designed and implemented. Methods were provided to imitate user actions in the context of a simulation, in order to provide realistic demonstrations. A new approach to authoring instruction using templates was developed. A number of basic templates for instructional interaction were developed, and simple dialog user interfaces were produced for authoring lesson specifications interactively. Classic VIVIDS was also revised so that it could export lessons for iRides using these new templates. The iRides Instructional Document Object Model (DOM) was made World Wide Web Consortium (W3C) compliant, so that lesson specifications can be edited with third-party tools and software modules. An improved method for presenting instructional text in the context of a simulation was developed.

- *Instructional authoring.* Recent work has produced a much improved instructional authoring system for developing lesson specifications for distance learning in the context of simulations. Authors can edit and test lessons interactively in iRides Author.

Earlier Results

Previously, we tested our simulation-centered assessment framework in two application areas: (a) USMC rifle marksmanship, and (b) Navy Engineering Duty Officer (EDO) risk management.

Application 1: USMC Rifle Marksmanship. The initial focus of the assessment and instructional research was on rifle marksmanship in collaboration with USMC Weapons Training Battalion (WTBN) at Stone Bay, NC, and with the WTBN at Quantico, VA. Simulations and simulation-centered training modules for marksmanship training were developed using CCT's iRides authoring and delivery systems. iRides and iRides Author, originally developed with ONR support, were enhanced and extended in the KMT project, and simulation-based training modules were developed for the Battlesight Zero procedure and for training on shot group analysis, creating the proper sight picture, and databook usage.

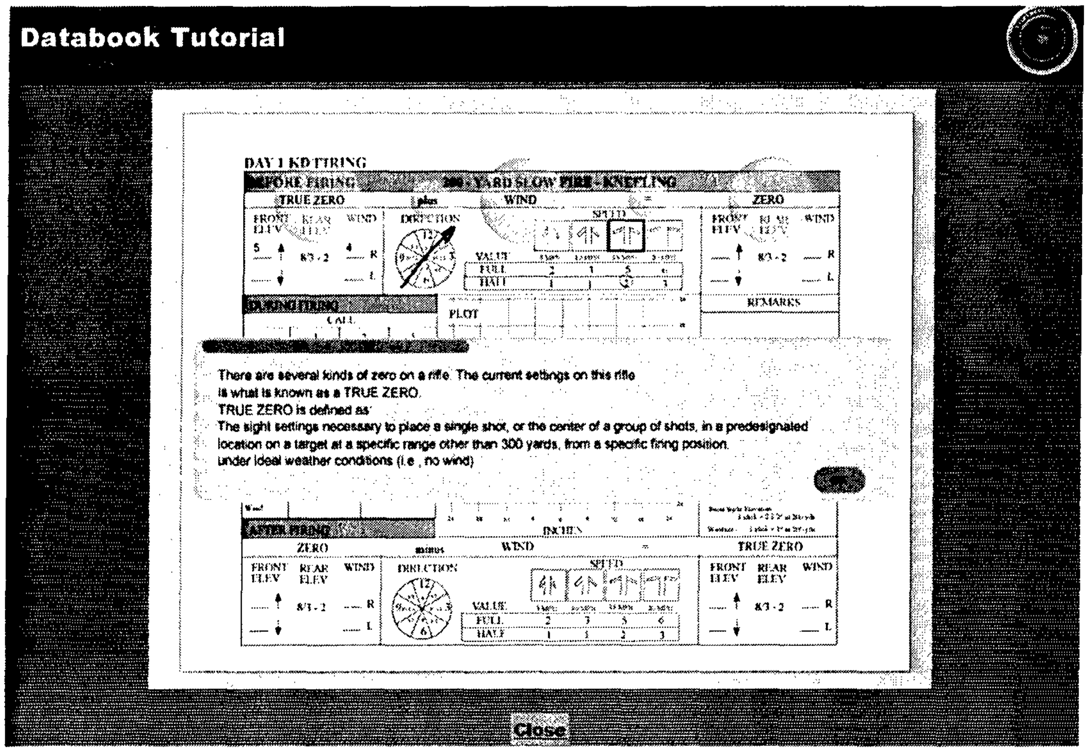
Background

Reducing the number of Marines who fail rifle marksmanship qualification is an important goal for the USMC. Identification of Marines likely to fail qualification, and subsequent remediation in a DL format, would result in considerable cost savings in terms of training time, time away from units, and rounds expended on the range.

Research

Training modules. Four distance learning modules on elements of marksmanship knowledge were developed: databook usage, sight picture, battlesight zero (BZO), and shot group analysis. These training resources provide an interactive graphical environment for Marines to refresh their knowledge about the correct use of the databook and related concepts in marksmanship.

The databook usage module presents a structured walkthrough of the use of the databook for Marines. It begins by reviewing the meaning of True Zero. (See Figure 4, below.) It guides the Marine to enter the weapon's current sight settings in the True Zero section in the upper left corner of the databook page. If the student makes an error, the module points out the error and guides the Marine to enter the correct values. The training module then guides the Marine to consider current wind conditions, and to correctly estimate the sight adjustment that those conditions call for. It then guides the completion of the Zero section of the databook page.



There are several kinds of zero on a rifle. The current settings on this rifle is what is known as a TRUE ZERO.
TRUE ZERO is defined as:
The sight settings necessary to place a single shot, or the center of a group of shots, in a predesignated location on a target at a specific range other than 300 yards, from a specific firing position, under ideal weather conditions (i.e., no wind)

Figure 4. Zero reflects current wind conditions.

The learner is also reminded that actual shot positions are to be recorded in the Plot section of the databook page, in the center area. After a group of three shots has been fired, the Marine is guided to enter any necessary adjustments at the fourth shot portion of the During Firing section. If incorrect values are entered, the Marine is given guidance on how to determine what values should be entered in this section. The module proceeds to guide the trainee on how to complete the process of sighting a weapon with the aid of the databook. Three related modules were also developed that contribute to training on how to use the databook effectively.

Application 2: Risk Management. A second application area for assessment and instructional research was risk management, an element of the Navy Engineering Duty Officer (EDO) School's acquisition training at Port Hueneme, CA. CRESST developed and pilot tested automated scoring techniques for a procedural knowledge format. In collaboration with CRESST scientists, CCT designed a tool for exploring the utility of possible decisions in a complex environment. This software tool, developed in iRides, can be used both to teach about decision making and to aid in the decision-making process.

Background

Successful project management in the acquisition process is dependent on how well an EDO is able to apply risk management principles. An important component of risk management is decision making that is based on thorough exploration of the problem space—the consideration of the relevant events likely to occur, the impact of those events, and an overall evaluative judgment of acceptable risk in the context of schedule, cost, and performance constraints.

Research

Measuring problem solving knowledge. Our primary focus was on developing a method to support the measurement of problem solving knowledge. Risk management was selected as the application area because of its problem solving complexity. We developed a general template for decision making based on the expert map. The general template is composed of environment variables, domain variables, options being considered, the analysis of the variables, and the final selection. In terms of risk management, the domain variables are cost, schedule, and performance. In the particular risk management scenario we tested, specific values were created for environment (requirements, regulations, legal) as well as for the different options EDOs had to consider. Preliminary findings suggest that the knowledge mapping measures show variation across individuals, and that this variation corresponds with experts' (i.e., instructors') evaluation of the knowledge map.

Decision Aid for Engineering Duty Officers. To support the EDO's analyses of risk management problems, a series of applications were developed that implement a multiattribute model of utility estimation for decision making. For a given type of decision, users can determine the attributes of the utility of outcomes. They can weight these attributes according to their contribution to outcome utility. For each event that might result from a decision, they can estimate the probability of each possible outcome of that decision. They can also estimate the attribute values for each of these outcomes.

Three specific examples of the application of this tool to decisions discussed in the EDO Basic Course were developed. In order of increasing complexity, these applications are:

- Choosing a restaurant;

- Selecting a digital camera; and
- Handling a *Refueling at Sea System* procurement problem, shown in Figure 5.

In addition, an empty version of the decision aid tool was developed for creating new decision trees from scratch.

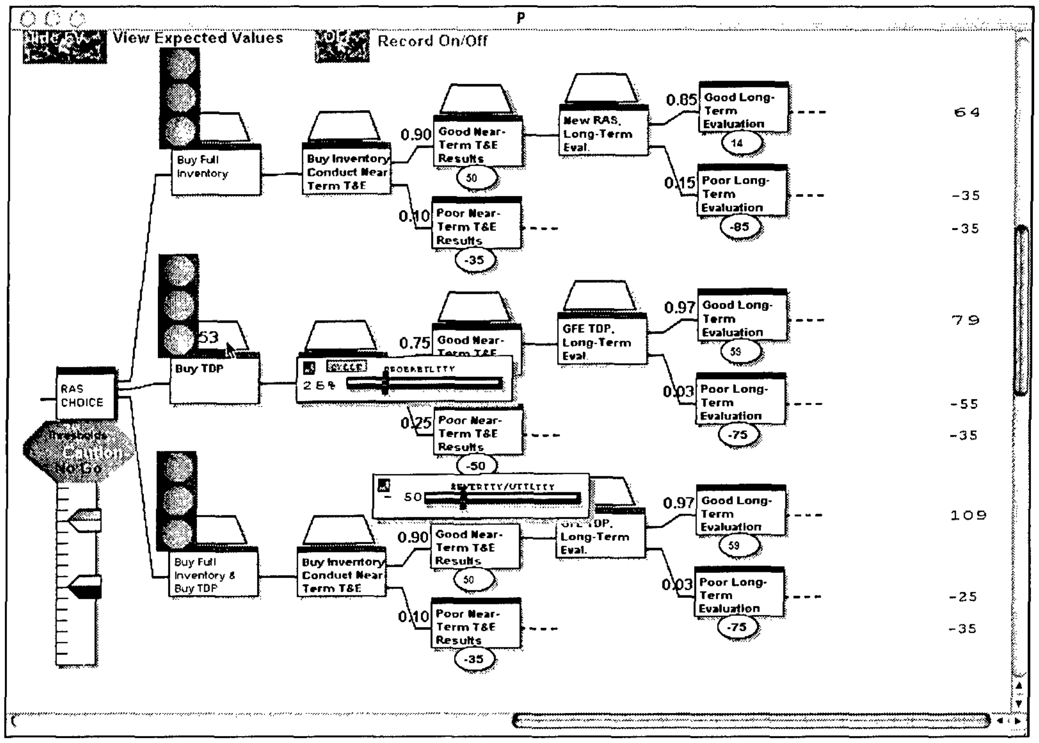


Figure 5. One view of the Refueling at Sea decision.

Technical Approach for This Project

Objectives. The overall goals of this effort were to apply the best scientific knowledge for improving the effectiveness of Navy and Marine distance learning applications through the tools used to make design and delivery choices, and to explore factors relevant to development for operational training environments in Naval Air and Surface Forces and in the Marine Corps. To achieve the Capable Manpower FNC objectives, the specific objectives of the CCT sub-award were:

1. To analyze a Navy training requirement that would benefit from the application of simulation-based pedagogy
2. To design, prototype, and test an instructional simulation system for that Navy subject matter, that can be delivered using the iRides technology

3. To integrate the simulation training system with the CRESST knowledge mapper assessment system.

Products of the effort are:

- A simulation-based instructional application for the Navy
- Knowledge mapper integration with iRides

The period of performance for the effort covered 12 months. (All CCT funds were expended before the contract extension for this supplement was granted.) The University of Southern California Center for Cognitive Technology worked to support the UCLA Center for Research on Evaluation, Standards, and Student Testing (CRESST), the prime contractor throughout this period. Work was segmented into three subtasks:

1. Experimental development of prototype Navy training subject matter
2. Research into new approaches for simple authoring of simulation-based assessments
3. Integration of the iRides delivered simulation training system with a CRESST knowledge mapper for the domain

The following three sections provide detailed descriptions of the work undertaken for these tasks.

Prototype Simulation Training with Navy Subject Matter

In initial discussions with the Navy Surface Warfare Officers School (SWOS) faculty about how officers could be assessed for relevant knowledge in the Department Head course, we found at first that instructors were unable to readily imagine assessing in any other way than with short-answer and multiple-choice question tests. To break through this conceptual block, we asked them whether they ever sketched situations on their white boards during class. Instructors identified two major (that is, particularly important) subject areas that they used extensive board diagramming to present in class. These topics are Air Defense (especially in carrier groups), and battle group communications and data systems structuring.

We explored both of these subject areas with the instructors with an eye to developing simulations that could be used to assess student competence in these areas. The Air Defense task is a complex one, in which there are many possible correct and incorrect actions in particular scenarios. This task seemed to be particularly well-suited to simulation-based assessment. We were able to construct a rough partial prototype of an Air Defense planning environment. When we showed the instructors that it would be possible to assess an officer's knowledge by asking him or her to position assets appropriately in several different threat contexts, they immediately recognized the potential of this approach and began suggesting features that would make it especially valuable to them. In the prototype, a student would be presented with the available resources stationed in a line. See Figure 6.

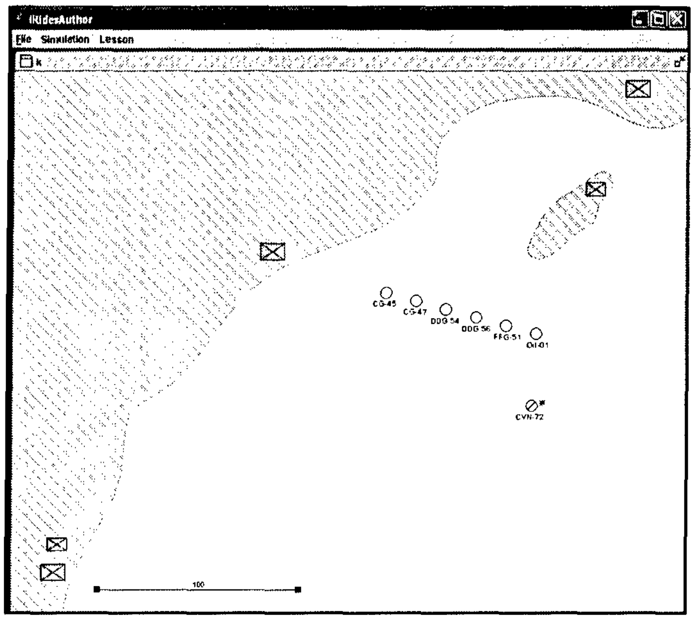


Figure 6. Initial prototype interface.

The student would then examine threat information, and would use the mouse to drag the icons that represented his resources (the ships of a carrier group) to establish an appropriate defense against air attack for the mission essential unit(s) of the group. The positioning of the ships and, eventually, the assignment of roles to the ships, would be automatically evaluated using rules given to us by the SWOS instructors.

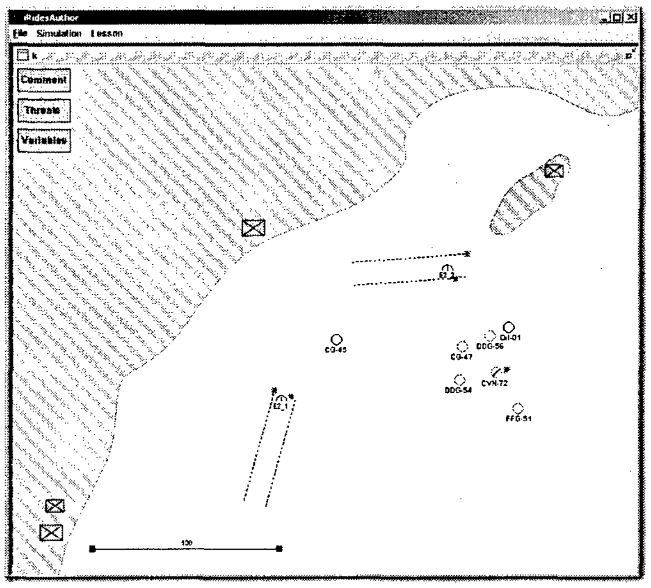


Figure 7. A possible student plan for Air Defense.

This work served as the basis for additional research and development efforts on simulation-based assessment in the context of surface warfare officer training, carried out under additional ONR and NETC funding.

Research on Novel Interfaces for Simple Simulation-based Assessment Development

A new simulation-authoring tool, initially called *Yasc* (for *Yet Another Simulation Constructor*), was developed for use in this study. Participants could use this software application to build simple graphical simulations to demonstrate their understanding of how something works. For example, a participant could build a simulation of how the various signal lights at an intersection are coordinated, and how they control traffic. Or the participant could construct a simulation of how rifle marksmanship is affected by breathing during firing.

Yasc has two major modes of use: composition mode, in which the participant builds an interactive graphical simulation, and demonstration mode, in which the participant uses a constructed simulation to show his or her knowledge about how something works. Participants will build simulations primarily by carrying out three types of actions:

- Selecting and positioning graphical shapes, images, and complex graphic objects

- Assigning behaviors to graphical objects
- Specifying what actions (e.g., user actions or simulation events) can initiate behaviors.

The user interface provides three libraries: one for objects (graphics), one for behaviors, and one for actions. See Figure 8, Figure 9, and Figure 10.

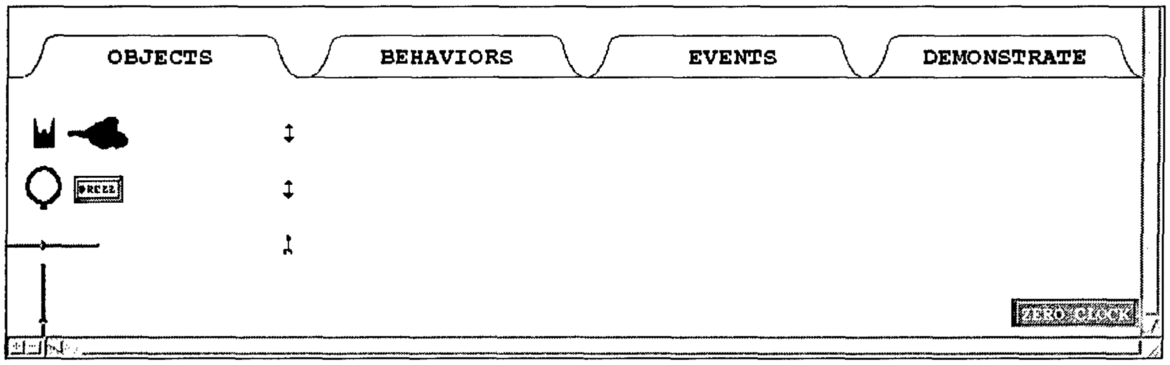


Figure 8. Object library.

The objects in this library include some general shapes, such as line segments, connected line segments, and lines with arrowheads. The library also has a number of objects related to rifle marksmanship, including graphics for a rifle front sight and a rear sight, a target, etc.

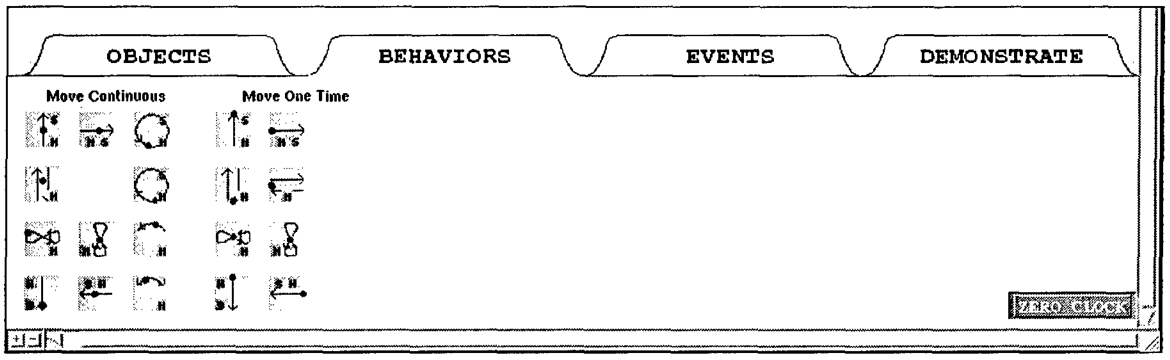


Figure 9. Behavior library.

The behaviors shown in the above behavior library include repetitive moves of several types: up and down, left and right, circular, figure eight, on an arc, and so on. It also includes similar movement behaviors that are not continuous, but rather apply only from one to some fixed number of times.

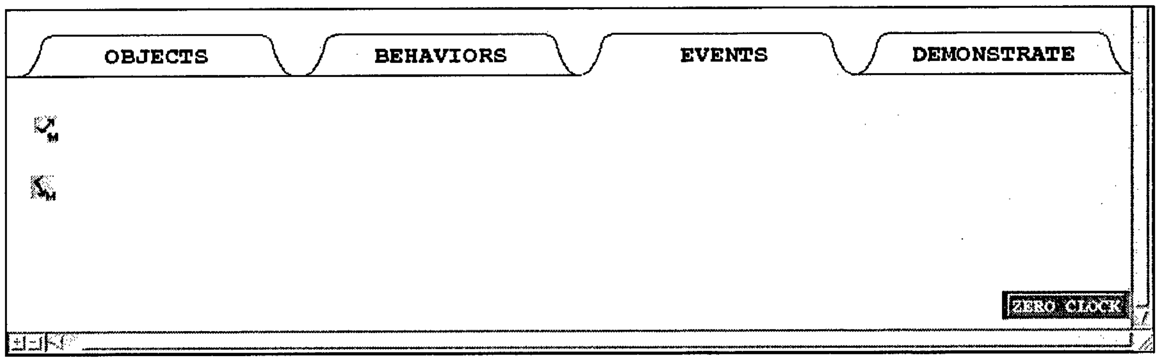


Figure 10. Action library.

This simple action library shows only *upclick* and *downclick* action elements.

To create a behavior that does not have to be triggered by an action, all the participant has to do is drag the icon for the desired behavior to a target object. Once the behavior is dropped on the object, it begins exhibiting the specified behavior.

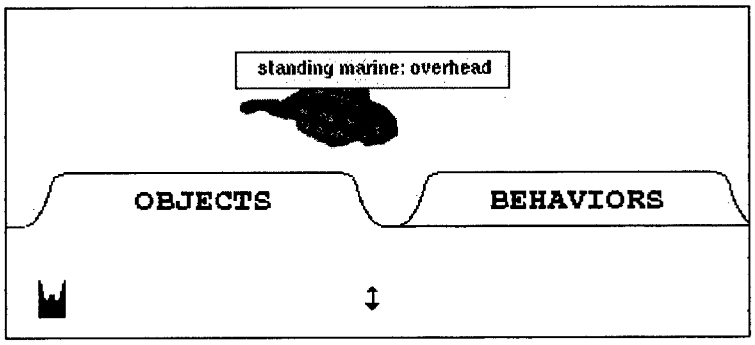


Figure 11. Dragging out an object.

In Figure 11, a participant drags an image of a Marine (viewed from overhead) into the simulation construction space. Some objects are automatically scaled to a larger size when they are placed in the simulation scene. As entities are dragged in construction mode, their names appear in an associated label.

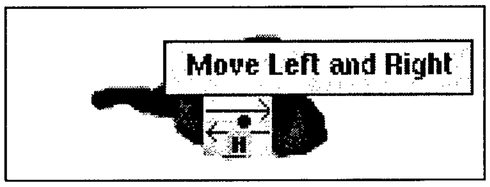


Figure 12. Dropping a behavior on an object.

When a user drops a behavior on an object, the object begins exhibiting the behavior. Here, the Marine moves left and right repeatedly as soon as the behavior is released on top of it.

Actions can be placed to trigger behaviors. First, the user must drag the action to the object that is the target of the action. In the case of an *upclick* action, this means the object that the upclick is to take place in. In Figure 13, the participant has placed two new objects on the simulation scene: a front sight (at the left of Figure 13) and a button. The button has been labeled "exhale." The user has also placed a "Move up one time" behavior on the front sight. (When this behavior is first placed on the front sight, the sight slowly moves up over a few seconds, under the control of the behavior. Because this is not a continuous behavior, the front sight does this only once.) The user wants the behavior of moving up to be triggered when the "exhale" button is pressed and released. So the upclick action is dragged to the "exhale" button. The action automatically connects itself to the button when it is dropped there. Now the user specifies the effect of the action by dragging the arrow (shown in pink, before it is connected, in Figure 13, and red, as it is being dragged in Figure 14) to the behavior that should result.

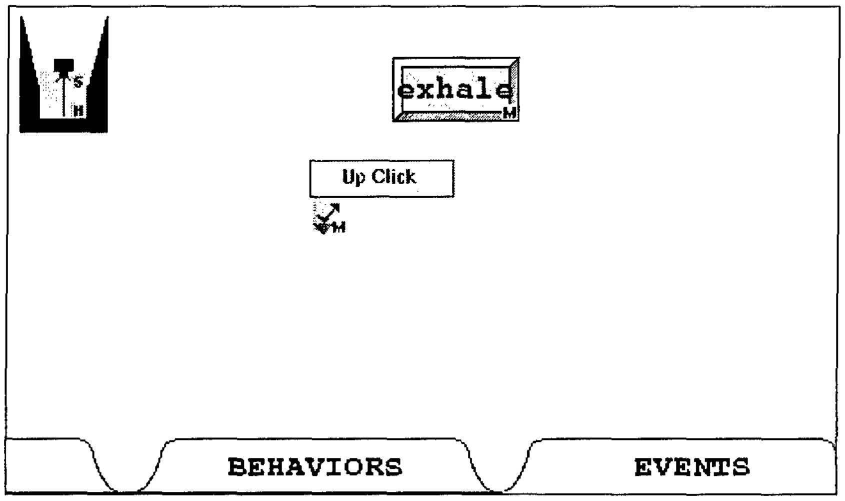


Figure 13. Composing a simple interactive simulation

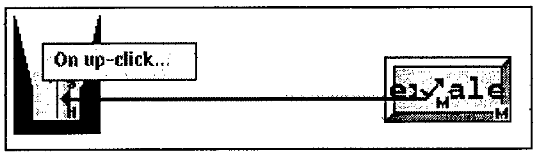


Figure 14. Specifying the behavior that results from an action.

When the behavior is specified, the arrow that connects the action element to the affected behavior is shown in green.

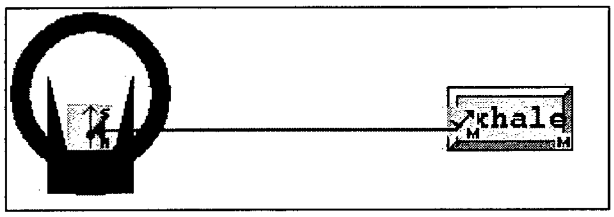


Figure 15. The construction mode view of a simple working simulation.

In Figure 15, the participant has also placed a rear sight graphic. This local context will make the movement of the front sight more visually apparent.

Finally, in demonstration mode, the user can show the working simulation that demonstrates his or her understanding of the system. In demonstration mode, the behavior elements and action elements are invisible, but they are what determine the interactive behavior of the simulation that was constructed.

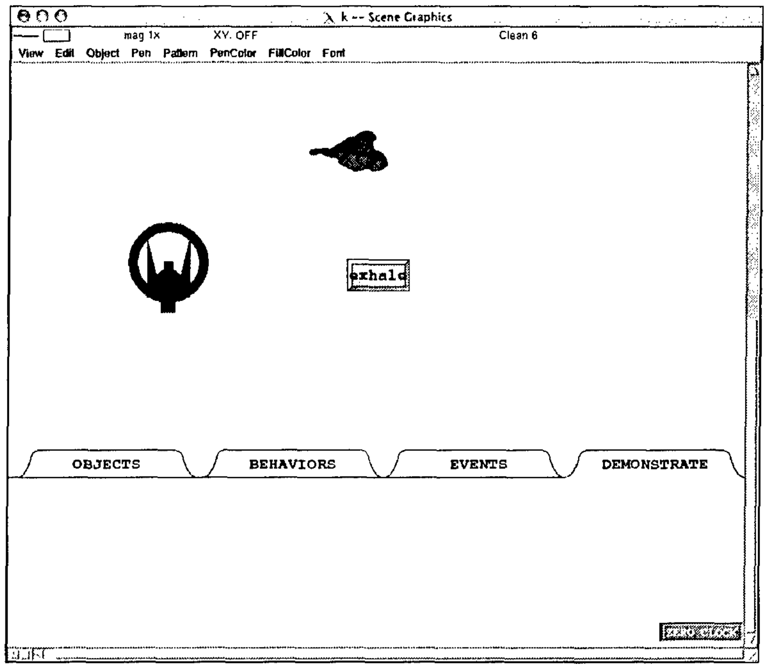


Figure 16. The simulation in demonstration mode

The simple interface for authoring simulation behaviors and interactivity by dragging and dropping was enhanced through the addition of cloning, so that multiple instances of objects, behaviors, and actions can be created on the fly by authors who might find the full iRides Author development environment too complex. This new environment was called SAT, for Simulation Assessment Tool.

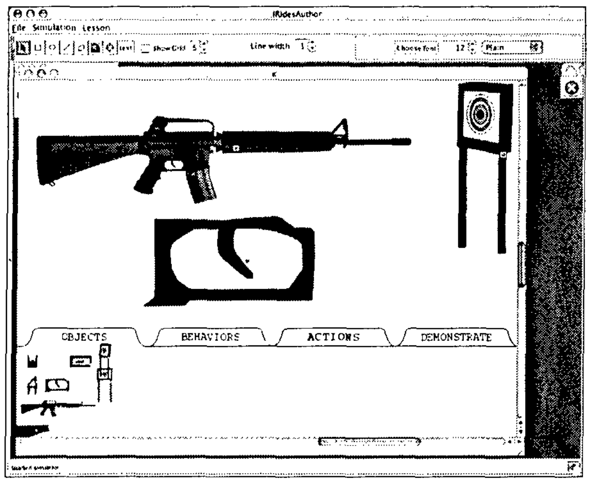


Figure 17. The SAT interface.

A simple version of SAT with only three shapes, three behaviors, and two action types was developed to be used to teach students about how the SAT environment works. See the figure below.

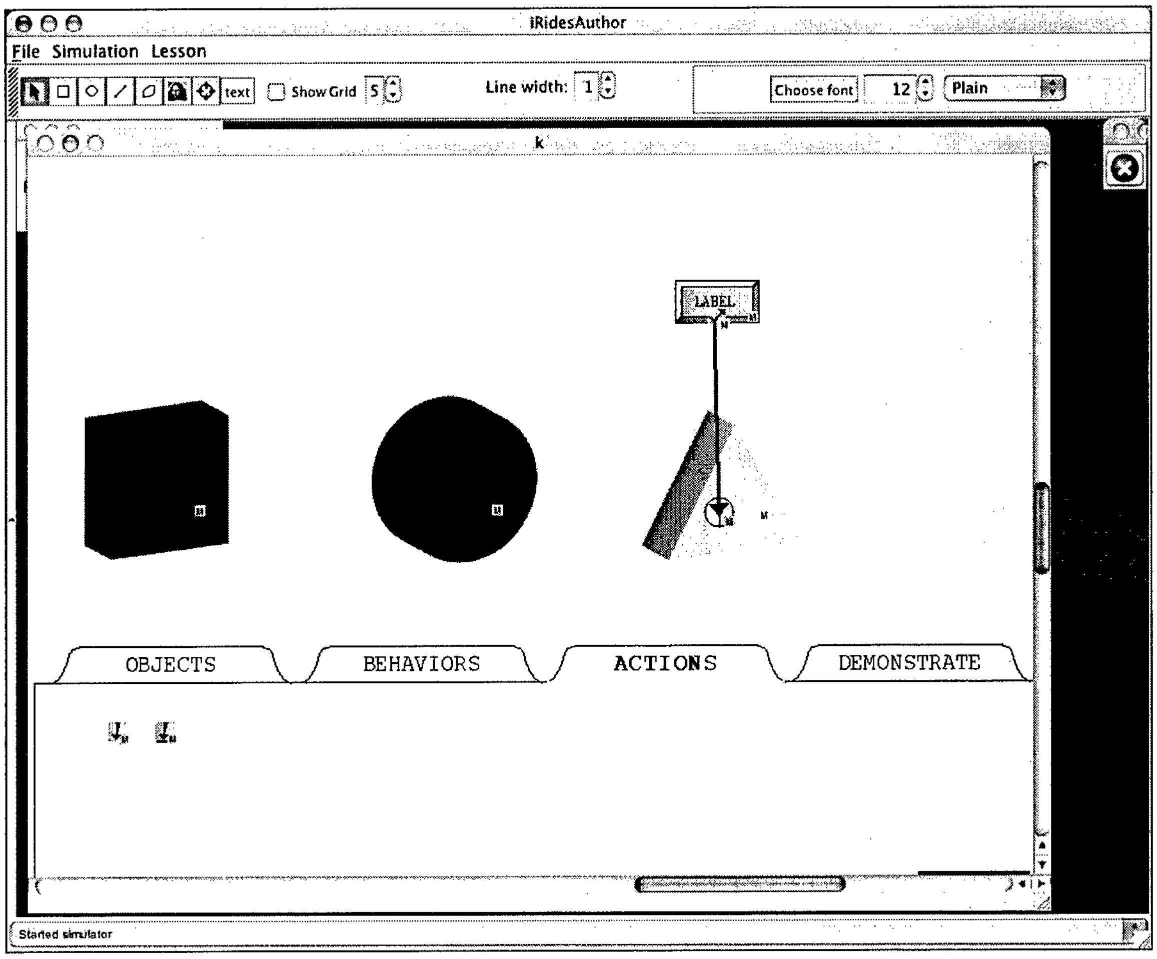


Figure 18. The SAT learning interface.

These experiments on simple simulation assessment authoring tools hint at the potential for considerable generality and simplicity in a simulation-authoring tool. Although there are many challenges to making this a general tool for rapid assessment of behavioral knowledge, future research is clearly merited.

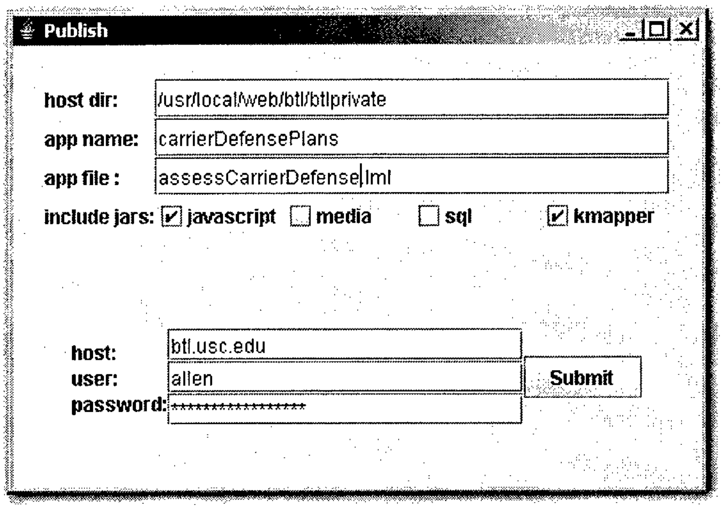
A second lesson from the experiences of developing YASC and SAT is that the iRides Author environment is one that offers sufficient power to support the development of new experimental authoring systems simply using the powerful features of iRides.

Integration of iRides Training with Knowledge Mapper Assessment

One goal in integrating the CRESST Knowledge Mapper with iRides was to support an optional packaging of the two products as one. For those applications that make use of one or the other, it should be possible to use them as separate products, without modifying source code and recompiling. This makes it possible to deliver lighter weight applications, and, in particular, lightweight applets when the functionality of both products is not required in a single application.

In iRides courses, scripts written in an XML language called Lesson Markup Language contain directives about what simulations should be used, what the student should be asked to do, how the student's actions should be judged for correctness, etc. These scripts are text files with the extension ".lml". One type of directive that an LML file can contain is to specify that media content like sound files or video should be played for the student. The low-level work of rendering the sound and/or video data is carried out by routines in a Java media package that is encapsulated in a .jar file. If a particular course, lesson, or assessment does not make use of such media, then the applet need not include that Java media file. We decided to incorporate the knowledge mapper into iRides in a similar way. The mapper jar file need not be incorporated in an iRides applet if the mapper will not be invoked for that application.

One of the ancillary tools that is provided with iRides is an *ant* script file ("build.xml") for carrying out a variety of frequently needed tasks. (*ant* is a Java program that is somewhat like *Make*. However, instead of being a scripting system that is extended with shell-based commands, Ant is extended using Java classes. The *ant* configuration files are XML-based, specifying a target tree where various tasks get executed. One of the major advantages to using *ant* is that the scripts in build.xml run identically on machines running under different operating systems.) The iRides version of build.xml is attached to this report as the Appendix.


Publish

host dir: /usr/local/web/btl/btlprivate

app name: carrierDefensePlans

app file : assessCarrierDefense.lml

include jars: [✓ javascript] media sql [✓ kmapper]

host: btl.usc.edu

user: allen

password: ******************

Figure 19. An interface for building iRides content with optional elements.

Two of the *ant* scripts from the iRides build.xml are *publish* and *publishapp.* These build scripts create executable packages of iRides content. Figure 19 shows the simple interface for specifying the contents of such a package. Four .jar files can be selected for inclusion, using the check boxes that appear before their names.

- *javascript*—This provides a Javascript interpreter as part of the LML execution system. Almost every instructional or assessment script makes extensive use of Javascript, which is an easily learned, easily read, and easily debugged programming language. In most LML lessons or assessment specifications, it is primarily used to invoke selected features of the delivery environment.

- *media*—The media jar provides the underlying capabilities that let iRides lesson files present sound and video resources to students or trainees.

- *sql*—An sql jar can be included if the script calls on the services of an sql database for storing or retrieving data during a student session.

- *kmapper*—The knowledge mapper jar provides the underlying capabilities of the CRESST knowledge mapper. The way in which these capabilities are invoked by an executing iRides LML file is described below.

iRides lesson scripts can open and interact with several different broad classes of complex user interfaces. These are simulations (called *models* in iRides), simulation graphics (*model views*), presentations (which can be simple text presentation interfaces or complex media players), and commands (usually buttons that provide meta-interactive capabilities, like quitting a lesson or asking for the answer to a question). The iRides name for the fifth broad class of invocable user interfaces is *requireEntry*. This class of objects is used to ask the students to do something other than interact with the simulation to demonstrate knowledge. A requireEntry object may be a simple dialog interface that asks a question and provides a text field into which the student types an answer. Other types of requireEntry objects pose multiple-choice questions, offer grids of cells—like a spreadsheet—or require that the student speak the answer to a question. The most natural role for the CRESST Knowledge Mapper is as a requireEntry object in iRides.

An LML script can invoke the mapper in this way.

```
<requireEntry entry="mapper">
      ...
</requireEntry>
```

The contents of the Mapper are determined by the data that are provided for it. In iRides, the actual data used by the mapper are specified in the properties file, *btl.prp*.

```
cresst.km.dataurl=http://tripoli.cse.ucla.edu:9090/
cresst.km.resourceurl=http://tripoli.cse.ucla.edu:8080/dev1/re
sources/
```

These URLs are passed to the knowledge mapper at run time to specify where mapper data can be retrieved from and stored.

iRides user interface objects, such as requireEntry elements, have a number of standard operations that can be invoked by LML scripts. These include positioning the interface on the screen, asking the object where its interface is positioned, clearing it, enabling and disabling it, making it appear and disappear, and so on. In addition, several new methods specific to mapper objects were implemented in iRides and are accessible to the script author as Javascript function calls placed in the LML text. These include

```
getConcepts()
getRelations()
getScores()
getExpertMapIDs()
saveMapper()
exitMapper()
```

The getScores() function makes it possible for an LML script to find out how well a student has done on a knowledge map. This information could then be used to determine what instructional experiences the student should next be offered. This means that performance in a knowledge map can be treated as an integrated part of an adaptive learning or assessment system coordinated through the use of iRides LML scripted specifications.

Additional support for application integration. It became clear that it would be useful if particular iRides training or assessment applications could be invoked with optional parameters that would be used by certain simulations or by particular training scripts. This would make it possible, for example, to invoke iRides while passing in a user ID that was relevant for the knowledge mapper.

The iRides player had previously been invoked with only a single parameter, which is the name of the data file to use. Typically, this would be an LML file that specified a course or lesson to be presented; that file, would, in turn, reference a simulation data file (a .jr file) that would be loaded to determine what simulation(s) would be presented. It was also possible to invoke iRides without an instructional specification, simply opening a .jr simulation specification. So, possible ways of starting an iRides application were, for example

```
java iRidesPlayer mainlesson.lml
```

and

```
java iRidesPlayer brakeSystem.jr
```

In the course of this project, iRides was modified so that one can pass values to the simulation at invocation time, using optional parameters. The additional arguments to iRidesPlayer must come in pairs, which consist of an attribute name followed by a value for that attribute. (If the parameter list ends prematurely, with an attribute name that is not followed by a value, the last parameter—the unmatched attribute name—is ignored.) The simulator treats these parameter pairs as the names and values of attributes of the .sys. scene. If an attribute of the same name as one in the parameter list already exists, it is given the specified value. If no such attribute exists, one is added and given the specified value.

One way that this new feature can be used is to invoke iRides with student ID information supplied by the invoking software. The simulation may then use this data in a variety of ways, including for reporting performance data, e.g.:

```
java iRidesPlayer mainlesson.lml studentID Lincoln_A module security
```

Here, the invoking agent has supplied the student name "Lincoln_A" and the name of the instructional module that is to be presented, "security."

Naturally, the iRides Author application can be used in exactly the same way, e.g.:

```
java iRidesAuthor edoDAT.jr helpLevel 2
```

In this case, the edoDAT simulation will be opened for authoring, with the attribute **`.sys.helpLevel`** set to 2.

Taken together, the integration of the Knowledge Mapper and the optional parameters of iRides can be used to create customized training solutions without requiring that developers modify the program code of either of these two applications.

References

Chung, G. K. W. K., Michiuye, J. K., Brill, D. G., Sinha, R., Saadat, F., de Vries, L. F., Delacruz, G. C., Bewley, W. L., & Baker, E. L. (2002). *CRESST Human Performance Knowledge Mapper System.* Los Angeles: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).

Chung, G. K. W. K., Sinha, R., de Souza e Silva, A. A., Cheak, A. M., Michiuye, J. K., Saadat, F., Bewley, W. L., & Baker, E. L. (2004). *CRESST Human Performance Knowledge Mapping Tool Authoring System.* Los Angeles: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).

Appendix: The ant *build.xml* Script for iRides Development Tasks

```
<project name="iRides" default="compile">
    <property name="sourcedir"              value="${basedir}/nv" />
    <property name="outputdir"              value="${basedir}/nvlib" />
    <property name="jarsdir"                value="${basedir}/nv/jars" />
    <property name="nvpackage"              value="nv"/>
    <property name="demospackage"           value="nvdemos"/>
    <property name="demosdir"               value="${basedir}/nvdemos"/>
    <property name="file"                   value="gte.lml"/>
    <property name="javacexcludes"
value="btl/svg/**,joshtest/**,btl/irbroker/**,btl/speechserver/**,btl/test
s/**,btl/sim/datatypes/procStatement.java,
btl/BCTAppletBuilder.java,btl/util/BctGui.java,btl/BtlStdBuilder.java,btl/
InSimApplet.java,btl/InSimLoader.java"/>

    <property name="javacplayerexcludes"
value="${javacexcludes},books/**,irides.java,btl/author/**,btl/graphics/vi
ewer2d/author/**,btl/AuthorBuilder.java,btl/IRidesAuthor.java,btl/util/Btl
AuthorGui.java, btl/IRidesAppletAuthor.java, btl/IDaveModel.java,
btl/IDavePlayer.java, btl/IDaveUserImpl.java,
btl/imp/instruction/editors/**,btl/Carroll.java"/>

    <property name="javacdaveexcludes"
value="${javacexcludes},irides.java,iridesP.java,btl/RuntimeBuilder.java,b
tl/util/**,btl/sim/**,btl/AbstractBuilder.java,btl/AbstractLoader.java,btl
/IRidesApplet.java,btl/IRidesPlayer.java,btl/BtlStdLoader.java,btl/Progres
sTracker.java,btl/IRidesApplet.java,btl/author/**,btl/graphics/**,btl/grap
hics/viewer2d/author/**,btl/AuthorBuilder.java,btl/IRidesAuthor.java,btl/u
til/BtlAuthorGui.java, btl/IRidesAppletAuthor.java, btl/IDaveModel.java,
btl/IDavePlayer.java, btl/IDaveUserImpl.java, btl/Carroll.java,
btl/imp/**"/>

    <property name="javacauthorexcludes"
value="${javacexcludes},brooks/**,btl/RuntimeBuilder.java,btl/IRidesPlayer
.java,btl/IRidesApplet.java,btl/util/BtlAppletGui.java,btl/util/RuntimeGui
.java,btl/util/BtlAppletGui.java,btl/util/BtlApplicationGui.java,btl/IDave
Model.java,btl/IDavePlayer.java,btl/IDaveUserImpl.java,
btl/Carroll.java"/>

    <property name="sshserver"              value="btl.usc.edu"/>
    <property name="sshuser"                value="allen"/>
    <property name="sshpassword"            value="YourPasswordHere"/>
    <property name="sshcvsname"             value="allen" />
    <property name="sshcvsroot"
value=":ext:${sshcvsname}@ratbeach.usc.edu:/cvsroot/irides" />
    <property name="releasedir"
value="/usr/local/web/btl/iRides/releases"/>
    <property name="releasejar"             value="testrelease.jar"/>
    <property name="publishurl"             value="http://btl.usc.edu/WS"
/>
    <property name="publishdir"
value="/usr/local/web/btl/WS"/>
```

```
    <property name="appleturl"
value="http://btl.usc.edu/applet" />
    <property name="publishappletdir"
value="/usr/local/web/btl/applets"/>
    <property name="appletcodebasedir"      value="someapplet"/>      <!--
someapplet -->
    <property name="applethref"             value="index.html"/>
    <property name="appletarg"              value="someapplet.lml"/>  <!--
someapplet.lml -->
    <property name="appletcontent"          value="somecontent.jar"/>  <!-
- someapplet.lml -->

    <property name="jnlpcodebasedir"        value="gizmo"/>          <!--
gizmo -->
    <property name="jnlphref"               value="gizmo.jnlp"/>       <!-
- gizmo.jnlp -->
    <property name="jnlptitle"              value="Nifty Gizmo"/>
<!-- nifty gizmo -->
    <property name="jnlpdesc"               value="This will solve all
your instructional needs!"/>
    <property name="jnlparg"                value="gizmo.lml"/>        <!-
- gizmo.lml -->
    <property name="jnlplml"                value="true"/>
    <property name="jnlpmedia"              value="true"/>
    <property name="jnlpmysql"              value="true"/>
    <property name="jnlpkmapper"            value="true"/>
    <property name="jnlpcontent"            value="gizmocontent.jar"/> <!-
- gizmocontent.jar -->

    <property name="apppublishdir"
value="/usr/local/web/btl/btlprivate"/>
    <property name="apppublishname"         value="somedemo"/>
    <property name="apppublishfile"         value="somedemo.lml"/>
    <property name="apppublishjars"         value="rhino15r41.jar jmf.jar
kmapper.jar"/>

    <property name="apppublishlibdir"       value="irides11-17-03"/>
    <property name="apppublishcontentdir"   value="content"/>
    <property name="fullreleasepath"        value="computedinpublish"/>

    <target name="jarapp">
      <jar destfile="${basedir}/app.jar">
   <fileset dir="${basedir}/nvlib"/>
   <fileset dir="${basedir}/nv" includes="rc/**"/>
      </jar>
    </target>

    <path id="classpath">
      <dirset dir="${outputdir}"/>
      <dirset dir="${sourcedir}"/>
      <fileset dir="${jarsdir}" id="jars">
        <include name="jaws.jar"/>
        <include name="rhino15r41.jar"/>
        <include name="rhino15r41-applet.jar"/>
        <include name="jmf.jar"/>
        <include name="kmapper.jar"/>
        <include name="mm.mysql-2.0.2-bin.jar"/>
        <include name="batik/batik-awt-util.jar"/>
        <include name="batik/batik-bridge.jar"/>
        <include name="batik/batik-css.jar"/>
        <include name="batik/batik-dom.jar"/>
        <include name="batik/batik-ext.jar"/>
        <include name="batik/batik-gvt.jar"/>
        <include name="batik/batik-parser.jar"/>
        <include name="batik/batik-script.jar"/>
        <include name="batik/batik-svg-dom.jar"/>
        <include name="batik/batik-swing.jar"/>
        <include name="batik/batik-util.jar"/>
        <include name="batik/batik-xml.jar"/>
        <include name="batik/irides-bakit.jar"/>
        <include name="batik/xerces_2_3_0.jar"/>
      </fileset>
    </path>

    <target name="sshexec">
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink /usr/local/web/btl/WS/latest-
release-dir"
                trust="true" outputproperty="sshoutput"/>
      <script language="beanshell">
        fullpath = project.getProperty("sshoutput");
        project.setProperty("crap",
fullpath.substring(fullpath.lastIndexOf("/")+1));
      </script>
      <echo>output:${sshoutput}</echo>
      <echo>crap:${crap}</echo>
      <echo></echo>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink /usr/local/web/btl/WS/latest-
release-jar"
                trust="true" outputproperty="sshoutput"/>
      <script language="beanshell">
        fullpath = project.getProperty("sshoutput");
        project.setProperty("garb",
fullpath.substring(fullpath.lastIndexOf("/")+1));
      </script>
      <echo>garb:${garb}</echo>
      <echo></echo>
    </target>

    <target name="ssh">
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink /usr/local/web/btl/WS/latest-
release-dir"
                trust="true" outputproperty="sshoutput"/>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cd
/usr/local/web/btl/iRides/releases;
```

```
                                                     zip garb.zip irides10-
20-03/irides10-20-03.jar irides10-20-30/jmf.jar irides10-20-03/kmapper.jar
irides10-20-03/rhino15r41.jar"
                trust="true" outputproperty="sshoutput"/>
      <echo>output:${sshoutput}</echo>

      <echo></echo>
    </target>

    <target name="startrecording">
      <record name="makerelease.log" action="start"/>
    </target>

    <target name="init">
       <tstamp>
   <format property="date" pattern="MM-dd-yy-mm-ss"/>
        <format property="releasedate" pattern="MM-dd-yy"/>
       </tstamp>
       <available property="sourcedirexist" value="true"
file="${sourcedir}"/>
       <available property="demosdirexist"   value="true"
file="${demosdir}"/>
       <echo>date:${date}</echo>
       <echo>releasedate:${releasedate}</echo>
       <echo></echo>
    </target>

    <target name="copysourcedir" if="sourcedirexist">
      <echo>copying existing directory ${sourcedir} to
${sourcedir}cpy${date}</echo>
      <echo></echo>
      <copy  toDir="${sourcedir}cpy${date}">
   <fileset dir="${sourcedir}" defaultexcludes="false">
   </fileset>
      </copy>
    </target>

    <target name="copydemosdir" if="demosdirexist">
      <echo>copying existing directory ${demosdir} to
${demosdir}cpy${date}</echo>
      <echo></echo>
      <copy  toDir="${demosdir}cpy${date}">
   <fileset dir="${demosdir}" defaultexcludes="false">
   </fileset>
      </copy>
    </target>

    <target name="clean" description="Removes all compiled java classes">
       <echo>deleting directory ${outputdir}</echo>
       <echo></echo>
       <delete dir="${outputdir}" />
    </target>

    <target name="prepare" description="Creates java class file
directory">
       <echo>making directory ${outputdir}</echo>
       <echo></echo>
```

```
        <mkdir dir="${outputdir}" />
    </target>

    <target name="compiledave" depends="prepare">
      <echo>compling dave  with the following values:</echo>
      <echo>srcdir:${sourcedir}</echo>
      <echo>destdir:${outputdir}</echo>
      <echo>exclude:${javacdaveexcludes}</echo>
      <echo></echo>
      <javac source="1.4" debug="on"  srcdir="${sourcedir}"
destdir="${outputdir}"
        classpathref="classpath" Excludes="${javacdaveexcludes}"/>
    </target>


    <target name="compileplayer" depends="prepare">
      <echo>compling player with the following values:</echo>
      <echo>srcdir:${sourcedir}</echo>
      <echo>destdir:${outputdir}</echo>
      <echo>exclude:${javacplayerexcludes}</echo>
      <echo></echo>
      <javac source="1.4" debug="on"  srcdir="${sourcedir}"
destdir="${outputdir}"
        classpathref="classpath" Excludes="${javacplayerexcludes}"/>
    </target>

    <target name="compileauthor" depends="prepare">
      <echo>compling author with the following values:</echo>
      <echo>srcdir:${sourcedir}</echo>
      <echo>destdir:${outputdir}</echo>
      <echo>exclude:${javacauthorexcludes}</echo>
      <echo></echo>
      <javac source="1.4" srcdir="${sourcedir}" destdir="${outputdir}"
        classpathref="classpath" Excludes="${javacauthorexcludes}"/>
    </target>

    <target name="compile" depends="prepare">
      <echo>compling both player and author with the following
values:</echo>
      <echo>srcdir:${sourcedir}</echo>
      <echo>destdir:${outputdir}</echo>
      <echo>exclude:${javacexcludes}</echo>
      <echo></echo>
      <javac source="1.4" srcdir="${sourcedir}" destdir="${outputdir}"
        classpathref="classpath" Excludes="${javacexcludes}"/>
    </target>

    <target name="builddave" depends="clean,prepare,compiledave">
    </target>

    <target name="buildplayer" depends="clean,prepare,compileplayer">
    </target>

    <target name="buildauthor" depends="clean,prepare,compileauthor">
    </target>

    <target name="build" depends="clean,prepare,compile">
    </target>

    <target name="checkoutnv" depends="init">
      <antcall target="copysourcedir"/>
      <delete dir="${sourcedir}"/>
      <cvs command="co" package="${nvpackage}" cvsRsh="ssh"
cvsroot="${sshcvsroot}" compression="true" compressionlevel="3"/>
      <echo>now need to build player (ant buildplayer), author (ant
buildauthor), or everything (ant build)</echo>
    </target>

    <target name="checkoutnvdemos" depends="init">
      <antcall target="copydemosdir"/>
      <delete dir="${demosdir}"/>
      <cvs command="co" package="${demospackage}" cvsRsh="ssh"
cvsroot="${sshcvsroot}" compression="true" compressionlevel="3"/>
    </target>

    <target name="checkout" depends="checkoutnv,checkoutnvdemos"/>

    <target name="updatenv">
      <cvs command="update -d" package="${nvpackage}" cvsRsh="ssh"
cvsroot="${sshcvsroot}" compression="true" compressionlevel="3"/>
      <echo>now need to build player (ant buildplayer), author (ant
buildauthor), or everything (ant build)</echo>
    </target>

    <target name="updatenvdemos">
      <cvs command="update -d" package="${demospackage}" cvsRsh="ssh"
cvsroot="${sshcvsroot}" compression="true" compressionlevel="3"/>
    </target>

    <target name="update" depends="updatenv, updatenvdemos"/>

    <target name="runplayer">
      <java classname="btl.IRidesPlayer"
            classpath="${user.dir};${jarsdir}/mm.mysql-2.0.2-bin.jar"
            classpathref = "classpath"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>

    <target name="launch">
      <java classname="launch"
            classpath="${user.dir};${jarsdir}/mm.mysql-2.0.2-bin.jar"
            classpathref = "classpath"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>

    <target name="rundave">
      <java classname="btl.IDavePlayer"
            classpath="${user.dir};${jarsdir}/mm.mysql-2.0.2-bin.jar"
            classpathref = "classpath"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>

<!--
    <target name="runplayer">
      <java classname="btl.IRidesPlayer"

classpath="${user.dir};${sourcedir};${outputdir};${jarsdir}/rhino15r41.jar
"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>
-->

    <target name="runauthor">
      <java classname="btl.IRidesAuthor"
            classpath="${user.dir};${jarsdir}/mm.mysql-2.0.2-bin.jar"
            classpathref = "classpath"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>

<!--
    <target name="runauthor">
      <java classname="btl.IRidesAuthor"

classpath="${user.dir};${sourcedir};${outputdir};${jarsdir}/rhino15r41.jar
"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>
-->
    <target name="runsvg" depends="compile">
      <java classname="btl.svg.SiRides"
            classpath="${user.dir};${jarsdir}/mm.mysql-2.0.2-bin.jar"
            classpathref = "classpath"
            fork="true"
            dir="${user.dir}">
   <arg value="${file}"/>
      </java>
    </target>

    <target name="jarrelease">
      <echo>making jar release with the following values:</echo>
      <echo>jarfile:${releasejar}</echo>
      <echo>basedir:${basedir}</echo>
      <echo>excludes:**/</echo>
      <echo>includes:rc/**, btl/**</echo>
      <echo></echo>
      <jar jarfile="${releasejar}" basedir="${basedir}" excludes="**/">
        <fileset dir="nv" includes="rc/**" />
        <fileset dir="nvlib" includes="btl/**" />
      </jar>
    </target>

    <target name="jarpublish">
      <echo>making jar publish with the following values:</echo>
      <echo>content:${user.dir}/${jnlpcontent}</echo>
      <echo>basedir:${user.dir}</echo>
      <echo>excludes:${jnlpcontent},**/CVS*,*.java</echo>
      <echo></echo>
      <jar jarfile="${jnlpcontent}" basedir="${user.dir}"
excludes="**/CVS/*, *.java"/>
    </target>

    <target name="jarapplet">
      <echo>making jar publish with the following values:</echo>
      <echo>content:${user.dir}/${appletcontent}</echo>
      <echo>basedir:${user.dir}</echo>
      <echo>excludes:${appletcontent},**/CVS*,*.java</echo>
      <echo></echo>
      <jar jarfile="${appletcontent}" basedir="${user.dir}"
excludes="**/CVS/*, *.java"/>
    </target>

    <target name="scppublish">
      <echo>scping publish with the following values:</echo>
      <echo>server:${sshserver}</echo>
      <echo>userid:${sshuser}</echo>
      <echo>remotedir:${publishdir}/${jnlpcodebasedir}</echo>
      <echo>dir:${user.dir}</echo>
      <echo>content:${jnlpcontent}</echo>
      <echo>jnlp file:${jnlphref}</echo>
      <echo></echo>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
                command="mkdir -p ${publishdir}/${jnlpcodebasedir}; cd
${publishdir}/${jnlpcodebasedir}; rm -rf * "
                trust="true"/>
      <!-- copyt latest release jar and need support jars to
apppublishdir/apppublishname/apppublishlibdir -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cp ${fullreleasepath}
${publishdir}/${jnlpcodebasedir};
                                                cd
${publishdir}/latest-release-dir;
                                                cp ${apppublishjars}
${publishdir}/${jnlpcodebasedir}"
                trust="true"/>
      <scp file="${user.dir}/${jnlphref}" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${publishdir}/${jnlpcodebase
dir}"/>
      <scp file="${basedir}/${jnlpcontent}" trust="true"
```

```
todir="${sshuser}:${sshpassword}@${sshserver}:${publishdir}/${jnlpcodebase
dir}"/>
    </target>


    <target name="scpapplet">
      <echo>scping publish applet  with the following values:</echo>
      <echo>server:${sshserver}</echo>
      <echo>userid:${sshuser}</echo>
      <echo>remotedir:${publishappletdir}/${appletcodebasedir}</echo>
      <echo>dir:${user.dir}</echo>
      <echo>content:${appletcontent}</echo>
      <echo>html file:${applethref}</echo>
      <echo>fullreleasepath:${fullreleasepath}</echo>
      <echo>jars:${apppublishjars}</echo>
      <echo></echo>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
                command="mkdir -p
${publishappletdir}/${appletcodebasedir}; cd
${appletpublishdir}/${appletcodebasedir}; rm -rf * "
                trust="true"/>
      <!-- copyt latest release jar and need support jars to
apppublishdir/apppublishname/apppublishlibdir -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cp ${fullreleasepath}
${publishappletdir}/${appletcodebasedir};
                                                   cd
${publishappletdir}/latest-release-dir;
                                                   cp ${apppublishjars}
${publishappletdir}/${appletcodebasedir}"
                trust="true"/>
      <scp file="${user.dir}/${applethref}" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${publishappletdir}/${applet
codebasedir}"/>
      <scp file="${basedir}/${appletcontent}" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${publishappletdir}/${applet
codebasedir}"/>
    </target>

    <target name="scprelease">
      <echo>cpping release with the following values:</echo>
      <echo>server:${sshserver}</echo>
      <echo>userid:${sshuser}</echo>
      <echo>remotedir:${releasedir}</echo>
      <echo>dir:${basedir}</echo>
      <echo>releasejar:${releasejar}</echo>
      <echo></echo>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"

                command="mkdir -p ${releasedir}; cd ${releasedir}; rm -rf
* "

                trust="true"/>
      <scp file="${basedir}/${releasejar}" trust="true"
           todir="${sshuser}:${sshpassword}@${sshserver}:${releasedir}"/>
      <scp trust="true"
           todir="${sshuser}:${sshpassword}@${sshserver}:${releasedir}">
        <fileset refid="jars"/>
      </scp>
    </target>

    <target name="publish">
      <script language="beanshell">
        codebasedir = new
JTextField(project.getProperty("jnlpcodebasedir"),20);
        dirpathlabel = new JLabel(project.getProperty("publishdir")+"/");
        jnlpfilename = new JTextField(project.getProperty("jnlphref"),20);
        title      = new JTextField(project.getProperty("jnlptitle"));
        description = new JTextField(project.getProperty("jnlpdesc"));
        argument     = new JTextField(project.getProperty("jnlparg"));
        content      = new JTextField(project.getProperty("jnlpcontent"));

        javascript   = new JCheckBox("javascript", true);
        media        = new JCheckBox("media", true);
        sql          = new JCheckBox("sql", true);
        kmapper      = new JCheckBox("kmapper",true);

        scpserver = new JTextField("btl.usc.edu", 20);
        scpuser = new JTextField(project.getProperty("sshuser"), 20);
        scppassword = new JPasswordField(20);
        frame = new JFrame("Publish");
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        //Layout the labels in a panel

        JPanel labelNamePane = new JPanel();
        labelNamePane.setLayout(new GridLayout(0, 1));
        labelNamePane.add(new JLabel("code base dir:"));
        labelNamePane.add(new JLabel("jnlp file name:"));
        labelNamePane.add(new JLabel("app title:"));
        labelNamePane.add(new JLabel("app description:"));
        labelNamePane.add(new JLabel("app argument:"));
        labelNamePane.add(new JLabel("content name:"));
        labelNamePane.add(new JLabel("include jars:"));

        JPanel labelFTPPane = new JPanel();
        labelFTPPane.setLayout(new GridLayout(0, 1));
        labelFTPPane.add(new JLabel("ftp server:"));
        labelFTPPane.add(new JLabel("ftp user:"));
        labelFTPPane.add(new JLabel("ftp password:"));


        //Layout the text fields in a panel
        JPanel buttons = new JPanel();
        buttons.setLayout(new GridLayout(0,4));
        buttons.add(javascript);
        buttons.add(media);
        buttons.add(sql);
        buttons.add(kmapper);

        JPanel garb = new JPanel();
```

```
garb.setLayout(new GridBagLayout());
garb.add(dirpathlabel);
garb.add(codebasedir);
JPanel fieldNamePane = new JPanel();
fieldNamePane.setLayout(new GridLayout(0, 1));
fieldNamePane.add(garb);
fieldNamePane.add(jnlpfilename);
fieldNamePane.add(title);
fieldNamePane.add(description);
fieldNamePane.add(argument);
fieldNamePane.add(content);
fieldNamePane.add(buttons);

JPanel fieldFTPPane = new JPanel();
fieldFTPPane.setLayout(new GridLayout(0, 1));
fieldFTPPane.add(scpserver);
fieldFTPPane.add(scpuser);
fieldFTPPane.add(scppassword);

//Put the panels in another panel, labels on left,
//text fields on right
JPanel contentNamePane = new JPanel();
contentNamePane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                              20, 20));
contentNamePane.setLayout(new BorderLayout());
contentNamePane.add(labelNamePane, BorderLayout.CENTER);
contentNamePane.add(fieldNamePane, BorderLayout.EAST);

JPanel contentFTPPane = new JPanel();
contentFTPPane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                              20, 20));

contentFTPPane.setLayout(new GridBagLayout());
contentFTPPane.add(labelFTPPane);
contentFTPPane.add(fieldFTPPane);
JButton submitbutton = new JButton("Submit");
contentFTPPane.add(submitbutton);

frame.getContentPane().setLayout(new GridLayout(0,1));
frame.getContentPane().add(contentNamePane);
frame.getContentPane().add(contentFTPPane);
frame.pack();
frame.setVisible(true);
boolean done = false;

actionPerformed(event){
  super.done = true;
};

submitbutton.addActionListener(this);
while(!done){
  Thread.sleep(50);
}
project.setProperty("jnlpcodebasedir", codebasedir.getText());
project.setProperty("jnlphref",  jnlpfilename.getText());
project.setProperty("jnlptitle",  title.getText());
project.setProperty("jnlpdesc",  description.getText());
```

```
            project.setProperty("jnlparg",  argument.getText());
            project.setProperty("jnlpcontent",  content.getText());
            String apppublishjars = "";
            if (javascript.isSelected()){
              project.setProperty("jnlplml","true");
              apppublishjars = "rhino15r41-applet.jar";
            }else {
              project.setProperty("jnlplml","false");
            }
            if (media.isSelected()){
              project.setProperty("jnlpmedia","true");
              apppublishjars = apppublishjars+" jmf.jar";
            }else {
              project.setProperty("jnlpmedia","false");
            }
            if (sql.isSelected()){
              project.setProperty("jnlpmysql","true");
              apppublishjars = apppublishjars+" mm.mysql-2.0.2-bin.jar";
            }else {
              project.setProperty("jnlpmysql","false");
            }
            if (kmapper.isSelected()){
              project.setProperty("jnlpkmapper","true");
              apppublishjars = apppublishjars+" kmapper.jar";
            }else {
              project.setProperty("jnlpkmapper","false");
            }
            project.setProperty("sshserver", scpserver.getText());
            project.setProperty("jnlpcodebasedir", codebasedir.getText());
            project.setProperty("sshuser",scpuser.getText());
            project.setProperty("sshpassword",scppassword.getText());
            project.setProperty("apppublishjars",apppublishjars);

       </script>
       <!-- read physical path of latest-release-dir and output to
outputproperty -->
       <sshexec  host="${sshserver}" username="${sshuser}"
                 password="${sshpassword}"
command="/usr/share/texmf/bin/readlink ${publishdir}/latest-release-dir"
                 trust="true" outputproperty="sshoutput"/>
       <!-- trim unwanted chars and file name from path -->
       <script language="beanshell">
         fullpath = project.getProperty("sshoutput").trim();
         project.setProperty("apppublishlibdir",
fullpath.substring(fullpath.lastIndexOf("/")+1));
       </script>
       <!-- read physical path of latest-release-jar and output to
outputproperty -->
       <sshexec  host="${sshserver}" username="${sshuser}"
                 password="${sshpassword}"
command="/usr/share/texmf/bin/readlink /usr/local/web/btl/WS/latest-
release-jar"
                 trust="true" outputproperty="sshoutput"/>
       <!-- trim unwanted chars from previous ssh output -->
       <script language="beanshell">
         fullpath = project.getProperty("sshoutput");
         project.setProperty("fullreleasepath", fullpath.trim());
```

```
      </script>
      <antcall target ="echojnlp"/>
      <antcall target ="jarpublish"/>
      <antcall target ="scppublish"/>
      <echo>url:${publishurl}/${jnlpcodebasedir}/${jnlphref}</echo>
    </target>


    <target name="makerelease" depends="startrecording, init">
      <script language="beanshell">
        releasename = "irides"+project.getProperty("releasedate");
        releasejarname = new JTextField(releasename+".jar",20);
        releasecvstag = new JTextField("rel_"+releasename,20);
        sshserver = new JTextField("btl.usc.edu", 20);
        releasedir = new
JTextField(project.getProperty("releasedir")+"/"+releasename,30);
        sshuser = new JTextField(20);
        sshpassword = new JPasswordField(20);
        frame = new JFrame("Make Release");
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        //Layout the labels in a panel
        JPanel labelNamePane = new JPanel();
        labelNamePane.setLayout(new GridLayout(0, 1));
        labelNamePane.add(new JLabel("release name:"));
        labelNamePane.add(new JLabel("release cvs tag:"));

        JPanel labelFTPPane = new JPanel();
        labelFTPPane.setLayout(new GridLayout(0, 1));
        labelFTPPane.add(new JLabel("ssh server:"));
        labelFTPPane.add(new JLabel("ssh remote dir:"));
        labelFTPPane.add(new JLabel("ssh user:"));
        labelFTPPane.add(new JLabel("ssh password:"));

        //Layout the text fields in a panel
        JPanel fieldNamePane = new JPanel();
        fieldNamePane.setLayout(new GridLayout(0, 1));
        fieldNamePane.add(releasejarname);
        fieldNamePane.add(releasecvstag);

        JPanel fieldFTPPane = new JPanel();
        fieldFTPPane.setLayout(new GridLayout(0, 1));
        fieldFTPPane.add(sshserver);
        fieldFTPPane.add(releasedir);
        fieldFTPPane.add(sshuser);
        fieldFTPPane.add(sshpassword);

       //Put the panels in another panel, labels on left,
       //text fields on right
        JPanel contentNamePane = new JPanel();
        contentNamePane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                                   20, 20));
        contentNamePane.setLayout(new BorderLayout());
        contentNamePane.add(labelNamePane, BorderLayout.CENTER);
        contentNamePane.add(fieldNamePane, BorderLayout.EAST);

        JPanel contentFTPPane = new JPanel();
        contentFTPPane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                                         20, 20));
        contentFTPPane.setLayout(new BorderLayout());
        contentFTPPane.add(labelFTPPane, BorderLayout.CENTER);
        contentFTPPane.add(fieldFTPPane, BorderLayout.EAST);

        frame.getContentPane().setLayout(new GridLayout(0,1));
        frame.getContentPane().add(contentNamePane);
        frame.getContentPane().add(contentFTPPane);
        JButton submitbutton = new JButton("Submit");
        frame.getContentPane().add(submitbutton);
        frame.pack();
        frame.setVisible(true);
        boolean done = false;

        actionPerformed(event){
          super.done = true;
        };

        submitbutton.addActionListener(this);
        while(!done){
          Thread.sleep(50);
        }
        project.setProperty("releasejar", releasejarname.getText());
        project.setProperty("releasecvstag", releasecvstag.getText());
        project.setProperty("sshserver", sshserver.getText());
        project.setProperty("releasedir", releasedir.getText());
        project.setProperty("sshuser",sshuser.getText());
        project.setProperty("sshpassword",sshpassword.getText());
      </script>

      <antcall target="jarrelease"/>


      <echo>creating cvs branch with following values:</echo>
      <echo>command:tag -b ${releasecvstag}</echo>
      <echo>package:${nvpackage}</echo>
      <echo>cvsroot:${sshsvsroot}</echo>
      <echo></echo>

      <cvs command="tag -b ${releasecvstag}" package="${nvpackage}"
cvsRsh="ssh" cvsroot="${sshcvsroot}" compression="true"
compressionlevel="3"/>

      <antcall target="scprelease"/>

      <echo>executing ssh commands with the following value:</echo>
      <echo>host:${sshserver}</echo>
      <echo>username:${sshuser}</echo>
      <echo>releasedir:${releasedir}</echo>
      <echo>releasejar:${releasejar}</echo>
      <echo></echo>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cd ${publishdir}; rm
latest-release-dir; ln -fs ${releasedir} latest-release-dir; rm latest-
release-jar; ln -fs ${releasedir}/${releasejar} latest-release-jar"
                trust="true"/>
     </target>
     <target name="publishapp">
      <script language="beanshell">
        apppublishdir = new
JTextField(project.getProperty("apppublishdir"));
        apppublishname    = new
JTextField(project.getProperty("apppublishname"));
        apppublishfile    = new
JTextField(project.getProperty("apppublishfile"));

        javascript  = new JCheckBox("javascript", true);
        media       = new JCheckBox("media", true);
        sql         = new JCheckBox("sql", false);
        kmapper     = new JCheckBox("kmapper",true);

        scpserver = new JTextField("btl.usc.edu", 20);
        scpuser = new JTextField(project.getProperty("sshuser"), 20);
        scppassword = new JPasswordField(20);

        frame = new JFrame("Publish");
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        //Layout the labels in a panel
        JPanel labelNamePane = new JPanel();
        labelNamePane.setLayout(new GridLayout(0, 1));
        labelNamePane.add(new JLabel("host dir:"));
        labelNamePane.add(new JLabel("app name:"));
        labelNamePane.add(new JLabel("app file :"));
        labelNamePane.add(new JLabel("include jars:"));

        JPanel labelFTPPane = new JPanel();
        labelFTPPane.setLayout(new GridLayout(0, 1));
        labelFTPPane.add(new JLabel("host:"));
        labelFTPPane.add(new JLabel("user:"));
        labelFTPPane.add(new JLabel("password:"));

        //Layout the text fields in a panel
        JPanel buttons = new JPanel();
        buttons.setLayout(new GridLayout(0,4));
        buttons.add(javascript);
        buttons.add(media);
        buttons.add(sql);
        buttons.add(kmapper);

        JPanel fieldNamePane = new JPanel();
        fieldNamePane.setLayout(new GridLayout(0, 1));
        fieldNamePane.add(apppublishdir);
        fieldNamePane.add(apppublishname);
        fieldNamePane.add(apppublishfile);
        fieldNamePane.add(buttons);

        JPanel fieldFTPPane = new JPanel();
        fieldFTPPane.setLayout(new GridLayout(0, 1));
        fieldFTPPane.add(scpserver);
        fieldFTPPane.add(scpuser);
        fieldFTPPane.add(scppassword);

       //Put the panels in another panel, labels on left,
//text fields on right
 JPanel contentNamePane = new JPanel();
 contentNamePane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                               20, 20));
 contentNamePane.setLayout(new BorderLayout());
 contentNamePane.add(labelNamePane, BorderLayout.CENTER);
 contentNamePane.add(fieldNamePane, BorderLayout.EAST);

 JPanel contentFTPPane = new JPanel();
 contentFTPPane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                               20, 20));

 contentFTPPane.setLayout(new GridBagLayout());
 contentFTPPane.add(labelFTPPane);
 contentFTPPane.add(fieldFTPPane);
 JButton submitbutton = new JButton("Submit");
 contentFTPPane.add(submitbutton);

 frame.getContentPane().setLayout(new GridLayout(0,1));

 frame.getContentPane().add(contentNamePane);
 frame.getContentPane().add(contentFTPPane);
 frame.pack();
 frame.setVisible(true);
 boolean done = false;

 actionPerformed(event){
   super.done = true;
 };

 submitbutton.addActionListener(this);
 while(!done){
   Thread.sleep(50);
 }
 project.setProperty("apppublishdir",  apppublishdir.getText());
 project.setProperty("apppublishname", apppublishname.getText());
 project.setProperty("apppublishfile", apppublishfile.getText());
 String apppublishjars = "";
 if (javascript.isSelected()){
   project.setProperty("jnlpml","true");
   apppublishjars = "rhino15r41-applet.jar";
 }else {
   project.setProperty("jnlpml","false");
 }
 if (media.isSelected()){
   project.setProperty("jnlpmedia","true");
   apppublishjars = apppublishjars+" jmf.jar";
 }else {
   project.setProperty("jnlpmedia","false");
 }
 if (sql.isSelected()){
   project.setProperty("jnlpmysql","true");
   apppublishjars = apppublishjars+" mm.mysql-2.0.2-bin.jar";
 }else {
   project.setProperty("jnlpmysql","false");
 }
 if (kmapper.isSelected()){
```

```
          project.setProperty("jnlpkmapper","true");
          apppublishjars = apppublishjars+" kmapper.jar";
        }else {
          project.setProperty("jnlpkmapper","false");
        }
        project.setProperty("sshserver", scpserver.getText());
        project.setProperty("sshuser",scpuser.getText());
        project.setProperty("sshpassword",scppassword.getText());
        project.setProperty("apppublishjars",apppublishjars);
      </script>
      <!-- create directory appppublishdir/appublishname -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="mkdir
${apppublishdir}/${apppublishname};"
                trust="true" />
      <!-- read physical path of latest-release-dir and output to
outputproperty -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink ${publishdir}/latest-release-dir"
                trust="true" outputproperty="sshoutput"/>
      <!-- trim unwanted chars and file name from path -->
      <script language="beanshell">
        fullpath = project.getProperty("sshoutput").trim();
        project.setProperty("apppublishlibdir",
fullpath.substring(fullpath.lastIndexOf("/")+1));
      </script>
      <!-- mkdir apppublishlibdir and content-->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="mkdir
${apppublishdir}/${apppublishname}/${apppublishlibdir};
                                                    mkdir
${apppublishdir}/${apppublishname}/content"
                trust="true" />
      <!-- read physical path of latest-release-jar and output to
outputproperty -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink /usr/local/web/btl/WS/latest-
release-jar"
                trust="true" outputproperty="sshoutput"/>
      <!-- trim unwanted chars from previous ssh output -->
      <script language="beanshell">
        fullpath = project.getProperty("sshoutput");
        project.setProperty("sshoutput", fullpath.trim());
      </script>
      <!-- copyt latest release jar and need support jars to
apppublishdir/apppublishname/apppublishlibdir -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cp ${sshoutput}
${apppublishdir}/${apppublishname}/${apppublishlibdir};
                                                    cd
${publishdir}/latest-release-dir;
                                                    cp ${apppublishjars}
${apppublishdir}/${apppublishname}/${apppublishlibdir}"
                trust="true"/>
      <scp file="${user.dir}/btl.prp" trust="true"
```

```
todir="${sshuser}:${sshpassword}@${sshserver}:${apppublishdir}/${apppublis
hname}"/>
      <zip destfile="${user.dir}/tempcontentverylongname.zip">

        <fileset dir="${user.dir}">
     <exclude name="btl.prp"/>
        </fileset>
      </zip>
       <scp trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${apppublishdir}/${apppublis
hname}/content">
        <fileset dir="${user.dir}">
     <include name="tempcontentverylongname.zip"/>
         </fileset>
      </scp>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cd
${apppublishdir}/${apppublishname}/content;
                                                     unzip
tempcontentverylongname.zip;
                                                     rm
tempcontentverylongname.zip"
                trust="true"/>
      <antcall target ="echomacrunscript"/>
      <scp file="${basedir}/macrun.tcsh" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${apppublishdir}/${apppublis
hname}"/>
      <scp file="${basedir}/macrunauth.tcsh" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${apppublishdir}/${apppublis
hname}"/>
      <antcall target ="echowinrunscript"/>
      <scp file="${basedir}/winrun.bat" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${apppublishdir}/${apppublis
hname}"/>
     <scp file="${basedir}/winrunauth.bat" trust="true"

todir="${sshuser}:${sshpassword}@${sshserver}:${apppublishdir}/${apppublis
hname}"/>
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cd
${apppublishdir}/${apppublishname};
                                                     chmod +x macrun.tcsh
macrunauth.tcsh winrun.bat winrunauth.bat"
                trust="true"/>
      <delete file="${user.dir}/tempcontentverylongname.zip"/>
      <!-- zip -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}" command="cd ${apppublishdir};
                                                     zip -r
${apppublishname}.zip ${apppublishname};
                                                     rm -r
${apppublishname}"
```

```
                trust="true"/>

    </target>

     <target name="publishapplet">
      <script language="beanshell">
        codebasedir = new
JTextField(project.getProperty("appletcodebasedir"),20);
        dirpathlabel = new
JLabel(project.getProperty("publishappletdir")+"/");
        htmlfilename = new
JTextField(project.getProperty("applethref"),20);
        argument    = new JTextField(project.getProperty("appletarg"));
        content    = new JTextField(project.getProperty("appletcontent"));

        javascript  = new JCheckBox("javascript", true);
        media       = new JCheckBox("media", true);
        sql         = new JCheckBox("sql", true);
        kmapper     = new JCheckBox("kmapper",true);

        scpserver = new JTextField("btl.usc.edu", 20);
        scpuser = new JTextField(project.getProperty("sshuser"), 20);
        scppassword = new JPasswordField(20);
        frame = new JFrame("Publish");
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        //Layout the labels in a panel
        JPanel labelNamePane = new JPanel();
        labelNamePane.setLayout(new GridLayout(0, 1));
        labelNamePane.add(new JLabel("code base dir:"));
        labelNamePane.add(new JLabel("html file name:"));
        labelNamePane.add(new JLabel("app argument:"));
        labelNamePane.add(new JLabel("content name:"));
        labelNamePane.add(new JLabel("include jars:"));

        JPanel labelFTPPane = new JPanel();
        labelFTPPane.setLayout(new GridLayout(0, 1));
        labelFTPPane.add(new JLabel("ftp server:"));
        labelFTPPane.add(new JLabel("ftp user:"));
        labelFTPPane.add(new JLabel("ftp password:"));

        //Layout the text fields in a panel
        JPanel buttons = new JPanel();
        buttons.setLayout(new GridLayout(0,4));
        buttons.add(javascript);
        buttons.add(media);
        buttons.add(sql);
        buttons.add(kmapper);

        JPanel garb = new JPanel();
        garb.setLayout(new GridBagLayout());
        garb.add(dirpathlabel);
        garb.add(codebasedir);
        JPanel fieldNamePane = new JPanel();
        fieldNamePane.setLayout(new GridLayout(0, 1));
        fieldNamePane.add(garb);
        fieldNamePane.add(htmlfilename);
        fieldNamePane.add(argument);
```

```
 fieldNamePane.add(content);
 fieldNamePane.add(buttons);

 JPanel fieldFTPPane = new JPanel();
 fieldFTPPane.setLayout(new GridLayout(0, 1));
 fieldFTPPane.add(scpserver);
 fieldFTPPane.add(scpuser);
 fieldFTPPane.add(scppassword);

//Put the panels in another panel, labels on left,
//text fields on right
 JPanel contentNamePane = new JPanel();
 contentNamePane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                                  20, 20));
 contentNamePane.setLayout(new BorderLayout());
 contentNamePane.add(labelNamePane, BorderLayout.CENTER);
 contentNamePane.add(fieldNamePane, BorderLayout.EAST);

 JPanel contentFTPPane = new JPanel();
 contentFTPPane.setBorder(BorderFactory.createEmptyBorder(20, 20,
                                                  20, 20));

 contentFTPPane.setLayout(new GridBagLayout());
 contentFTPPane.add(labelFTPPane);
 contentFTPPane.add(fieldFTPPane);
 JButton submitbutton = new JButton("Submit");
 contentFTPPane.add(submitbutton);

 frame.getContentPane().setLayout(new GridLayout(0,1));
 frame.getContentPane().add(contentNamePane);
 frame.getContentPane().add(contentFTPPane);
 frame.pack();
 frame.setVisible(true);
 boolean done = false;


 actionPerformed(event){
   super.done = true;
 };

 submitbutton.addActionListener(this);
 while(!done){
   Thread.sleep(50);
 }
 project.setProperty("appletcodebasedir", codebasedir.getText());
 project.setProperty("applethref",  htmlfilename.getText());
 project.setProperty("appletarg",  argument.getText());
 project.setProperty("appletcontent",  content.getText());
 String apppublishjars = "";
 if (javascript.isSelected()){
   project.setProperty("jnlplml","true");
   apppublishjars = "rhino15r41-applet.jar";
 }else {
   project.setProperty("jnlplml","false");
 }
 if (media.isSelected()){
   project.setProperty("jnlpmedia","true");
```

```
          apppublishjars = apppublishjars+" jmf.jar";
        }else {
          project.setProperty("jnlpmedia","false");
        }
        if (sql.isSelected()){
          project.setProperty("jnlpmysql","true");
          apppublishjars = apppublishjars+" mm.mysql-2.0.2-bin.jar";
        }else {
          project.setProperty("jnlpmysql","false");
        }
        if (kmapper.isSelected()){
          project.setProperty("jnlpkmapper","true");
          apppublishjars = apppublishjars+" kmapper.jar";
        }else {
          project.setProperty("jnlpkmapper","false");
        }
        project.setProperty("sshserver", scpserver.getText());
        project.setProperty("appletcodebasedir", codebasedir.getText());
        project.setProperty("sshuser",scpuser.getText());
        project.setProperty("sshpassword",scppassword.getText());
        project.setProperty("apppublishjars",apppublishjars);

      </script>
      <!-- read physical path of latest-release-dir and output to
outputproperty -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink ${publishdir}/latest-release-dir"
                trust="true" outputproperty="sshoutput"/>
      <!-- trim unwanted chars and file name from path -->
      <script language="beanshell">
        fullpath = project.getProperty("sshoutput").trim();
        project.setProperty("apppublishlibdir",
fullpath.substring(fullpath.lastIndexOf("/")+1));
      </script>
      <!-- read physical path of latest-release-jar and output to
outputproperty -->
      <sshexec  host="${sshserver}" username="${sshuser}"
                password="${sshpassword}"
command="/usr/share/texmf/bin/readlink /usr/local/web/btl/WS/latest-
release-jar"
                trust="true" outputproperty="sshoutput"/>
      <!-- trim unwanted chars from previous ssh output -->
      <script language="beanshell">
        fullpath = project.getProperty("sshoutput");
        project.setProperty("fullreleasepath", fullpath.trim());
      </script>
      <antcall target ="echojnlp"/>
      <antcall target ="jarapplet"/>
      <antcall target ="scpapplet"/>
<!--
      <echo>url:${appleturl}/${appletcodebasedir}/${applethref}</echo>
-->
    </target>

    <target name="echoall">
      <echoproperties/>
    </target>

    <target name="echo">
      <echo>basedir:    ${basedir}</echo>
      <echo>user.dir    ${user.dir}</echo>
      <echo>sourcedir:  ${sourcedir}</echo>
      <echo>outputdir:  ${outputdir}</echo>
      <echo>jarsdir:    ${jarsdir}</echo>
      <echo>cvsroot:    ${cvsroot}</echo>
      <echo>demosdir:   ${demosdir}</echo>
      <echo>file:       ${file}</echo>
    </target>

    <target name = "echomacrunscript">
      <echo>creating mac run script</echo>
      <echo></echo>
      <script language="beanshell">
        <![CDATA[
          StringBuffer buffer = new StringBuffer();
          buffer.append("#!/bin/tcsh"+"\n");
          buffer.append("set libdir =
./"+project.getProperty("apppublishlibdir")+"\n");
          buffer.append("set contentdir =
./"+project.getProperty("apppublishcontentdir")+"\n");
          buffer.append("set libjar =
"+project.getProperty("apppublishlibdir")+".jar"+"\n");
          String apppublishjars = project.getProperty("apppublishjars");
          StringBuffer tmpbuffer = new StringBuffer();
          tmpbuffer.append("${libdir}/");
          for(int i=0; i<apppublishjars.length(); i++){
             if (apppublishjars.charAt(i) == ' '){
               tmpbuffer.append(":${libdir}/");
             }else {
               tmpbuffer.append(apppublishjars.charAt(i));
             }
          }
          buffer.append("set supportjars = "+tmpbuffer.toString()+"\n");
          buffer.append("set localclasspath =
.:${libdir}/${libjar}:${supportjars}:${contentdir}"+"\n");
          project.setProperty("tmprunplayer", buffer.toString()+"java -
classpath ${localclasspath} btl.IRidesPlayer
${contentdir}/"+project.getProperty("apppublishfile")+"\n");
          project.setProperty("tmprunauthor", buffer.toString()+"java -
classpath ${localclasspath} btl.IRidesAuthor
${contentdir}/"+project.getProperty("apppublishfile")+"\n");
        ]]>
       </script>
        <echo file="macrun.tcsh">${tmprunplayer}</echo>
        <echo file="macrunauth.tcsh">${tmprunauthor}</echo>
    </target>

    <target name = "echowinrunscript">
      <echo>creating win run script</echo>
      <echo></echo>
      <script language="beanshell">
        <![CDATA[
          StringBuffer buffer = new StringBuffer();
          buffer.append("set
libdir=./"+project.getProperty("apppublishlibdir")+"\n");
          buffer.append("set
contentdir=./"+project.getProperty("apppublishcontentdir")+"\n");
          buffer.append("set
libjar="+project.getProperty("apppublishlibdir")+".jar"+"\n");
          String apppublishjars = project.getProperty("apppublishjars");
          StringBuffer tmpbuffer = new StringBuffer();
          tmpbuffer.append("%libdir%/");
          for(int i=0; i<apppublishjars.length(); i++){
             if (apppublishjars.charAt(i) == ' '){
               tmpbuffer.append(";$libdir%/");
             }else {
               tmpbuffer.append(apppublishjars.charAt(i));
             }
          }
          buffer.append("set supportjars="+tmpbuffer.toString()+"\n");
          buffer.append("set
localclasspath=.;%libdir%/%libjar%;%supportjars%;%contentdir%"+"\n");
          project.setProperty("tmprunplayer", buffer.toString()+"java -
classpath %localclasspath% btl.IRidesPlayer
%contentdir%/"+project.getProperty("apppublishfile")+"\n");
          project.setProperty("tmprunauthor", buffer.toString()+"java -
classpath %localclasspath% btl.IRidesAuthor
%contentdir%/"+project.getProperty("apppublishfile")+"\n");
        ]]>
       </script>
        <echo file="winrun.bat">${tmprunplayer}</echo>
        <echo file="winrunauth.bat">${tmprunauthor}</echo>
    </target>

    <target name="echojnlp">
      <echo>creating jnlp file:${user.dir}/${jnlphref}</echo>
      <echo></echo>
      <script language="beanshell">
      <![CDATA[
        StringBuffer buffer = new StringBuffer();
        buffer.append("<?xml version=\"1.0\" encoding=\"utf-8\"?>"+"\n");
        buffer.append("<jnlp spec=\"1.0+\" codebase=\""+
              project.getProperty("publishurl")+"/"+
              project.getProperty("jnlpcodebasedir")+
              "\" href=\""+project.getProperty("jnlphref")+"\">"+"\n");
        buffer.append("\t<information>"+"\n");

buffer.append("\t\t<title>"+project.getProperty("jnlptitle")+"</title>"+"\
n");
        buffer.append("\t\t<vendor>Behavioral Technology
Laboratories</vendor>"+"\n");
        buffer.append("\t\t<homepage>../index.html</homepage>"+"\n");
        buffer.append("\t\t<description kind=\"short\">"+
              project.getProperty("jnlpdesc")+
              "</description>"+"\n");
        //buffer.append("\t\t<offline-allowed/>"+"\n");
        buffer.append("\t</information>"+"\n");
        buffer.append("\t<security>"+"\n");
        buffer.append("\t</security>"+"\n");
        buffer.append("\t<resources>"+"\n");
        buffer.append("\t\t<j2se version=\"1.4\"/>"+"\n");
        //buffer.append("\t\t<jar href=\"../latest-release-jar\"/>"+"\n")
        buffer.append("\t\t<jar
href=\""+project.getProperty("apppublishlibdir")+".jar"+"\"/>"+"\n");
        buffer.append("\t\t<jar
href=\""+project.getProperty("jnlpcontent")+"\"/>"+"\n");
        String jarhref = null;
        for(int i=0; i<4; i++){
         switch(i){
          case 0:{
            if ("true".equals(project.getProperty("jnlplml"))){
              jarhref = "rhino15r41-applet.jar";
            }
            break;
          }case  1:{
            if ("true".equals(project.getProperty("jnlpmedia"))){
              jarhref = "jmf.jar";
            }
            break;
          }case 2:{
            if ("true".equals(project.getProperty("jnlpmysql"))){
              jarhref = "mm.mysql-2.0.2-bin.jar";
            }
            break;
          }case 3:{
            if ("true".equals(project.getProperty("jnlpkmapper"))){
              jarhref = "kmapper.jar";
            }
            break;
          }default:{
            buffer.append("ERROR in build.xml, echojnlp"+"\n");
          }
         }
         if (jarhref != null){
           buffer.append("\t\t<jar href=\""+jarhref+"\"
download=\"lazy\"/>"+"\n");
           jarhref = null;
         }

        }
        buffer.append("\t</resources>"+"\n");
        buffer.append("\t<application-desc main-
class=\"btl.IRidesPlayer\">"+"\n");
        buffer.append("\t\t<argument>");
        buffer.append(project.getProperty("jnlparg"));
        buffer.append("</argument>"+"\n");
        buffer.append("\t</application-desc>"+"\n");
        buffer.append("</jnlp>"+"\n");
        project.setProperty("garb",buffer.toString());
        //print(buffer.toString());
      ]]>
      </script>
      <echo file="${user.dir}/${jnlphref}">${garb}</echo>
      <echo>contents:</echo>
      <echo>${garb}</echo>
      <echo></echo>
      <xmlvalidate file="${user.dir}/${jnlphref}" lenient="true"
warn="true" />
    </target>
</project>
```